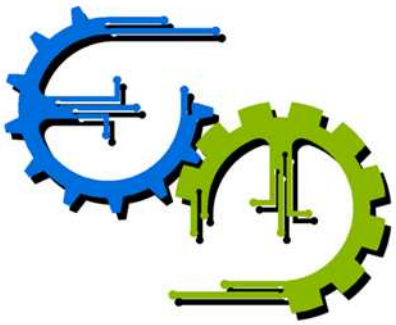
TRABALHO DE GRADUAÇÃO

Controle de Movimentação de Humanóide em Tempo Real por Teleoperação

Por,

Marcela Pinheiro de Carvalho

Brasília, julho de 2014



ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

Controle de Movimentação de Humanóide em Tempo Real por Teleoperação

Por,
Marcela Pinheiro de Carvalho

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Antonio Padilha Lanari Bó, ENE/UnB _____
Orientador

Prof. Geovany Araújo Borges, ENE/UnB _____
Examinador interno

Profa. Mariana Costa Bernardes , ENE/UnB _____
Examinadora interna

Brasília, julho de 2014

FICHA CATALOGRÁFICA

MARCELA PINHEIRO DE CARVALHO
Controle de Movimentação de Humanóide em Tempo Real por Teleoperação,

[Distrito Federal] 2014.

x, 49p., 297 mm (FT/UnB, Engenheira, Controle e Automação, 2014). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. NAO 2.Teleoperação
3. Quatérnions duais

I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

MARCELA, P. C., (2014). Controle de Movimentação de Humanóide em Tempo Real por Teleoperação. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n°$02, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 49p.

CESSÃO DE DIREITOS

AUTOR: Marcela Pinheiro de Carvalho

TÍTULO DO TRABALHO DE GRADUAÇÃO: Controle de Movimentação de Humanóide em Tempo Real por Teleoperação.

GRAU: Engenheira ANO: 2014

Marcela Pinheiro de Carvalho

Dedicatória

Dedico o meu trabalho aos meus pais, a minha irmã e a minha avó, pelo exemplo, pela educação e pelo carinho que sempre me proporcionaram.

Marcela Pinheiro de Carvalho

Agradecimentos

Inicialmente, gostaria de agradecer aos professores que participaram de toda a minha formação. Em especial agradeço ao meu orientador, o professor Antonio Padilha e ao meu co-orientador, Murilo Marinho, que me deram as ferramentas e os meios para a concretização do projeto. Cabe ainda agradecer aos membros da banca, o professor Geovany Borges e a professora Mariana, do LARA. Agradeço ainda ao professor José Camargo, Letícia e Gilmar por me orientarem em projetos de pesquisas no LTSD durante a minha graduação.

Gostaria de agradecer aos meus colegas e amigos. Desde o colégio, em que pude contar com a amizade e a inteligência da Héricka, da Jú e da Bia, em Anápolis, e da Samia, da Fê, da Nina e da Cari, em Brasília. À universidade, onde conheci meus grandes amigos, Guiga (sempre disponível para conversas, resolução de exercícios por telefone e um sushi em Paris), Tiaguinho e Bruno , que me apoiaram e me acompanharam em diversos trabalhos, em madrugadas de estudo e em momentos de crescimento pessoal.

Em especial, agradeço à Aninha pelos sonhos e desafios compartilhados, obrigada pelos ensinamentos e pelo exemplo de dedicação e compromisso. Agradeço ainda à Ana Clara, ao Levy, ao André, ao Gaucho, ao Zé e ao Pedro. Obrigada a todos os membros da Equipe DROID, em especial, ao Gi e ao Lipe, por me ensinarem tanto sobre a mecatrônica e sobre o desenvolvimento de robôs. Saindo do âmbito da mecatrônica, agradeço a todos os meus amigos do LPCI, o Ferreira, o Heider e o Lucas, com quem passei grande parte do meu tempo esse semestre.

Agradeço aos amigos que fiz na França, a Paula, o Heitor, o Ivan, o Matheus, a Pauline e os amigos da ENSTA Bretagne e da THALES. Com um carinho especial, agradeço ao meu namorado, Joffrey, por me apoiar sempre que precisei.

Finalmente, desejo agradecer à minha família, base de todo meu conhecimento e da minha educação. Agradeço por apoiarem meus sonhos e minhas escolhas, pelo exemplo de honestidade, persistência e amor. Obrigada à minha mãe, Suêrda, por me ensinar a sempre ter humildade e organização em tudo o que eu fizesse. Ao meu pai, Isaias, por me ensinar a amar o estudo e os desafios. À minha avó, Maria José, pela paixão pela França e pelo exemplo de mulher. À minha irmã, pela paciência e pelo carinho.

Marcela Pinheiro de Carvalho

RESUMO

Este trabalho se trata do desenvolvimento de um pacote em ROS (*Robot Operating System*), para o controle de movimentação de um humanóide por teleoperação. Validamos neste projeto a utilização do controle por cinemática inversa através de quatérnios duais. A plataforma robótica NAO apresenta-se como um *hardware* robusto para aplicação das teorias de controle aqui utilizadas. Para captura dos movimentos humanos, utilizamos o sensor Kinect. Os próximos capítulos apresentam a teoria e o desenvolvimento dessa iniciativa.

Palavras Chave: NAO, cinemática inversa, quatérnios duais, matriz de transformação, pseudo-inversa, jacobiana, controle, manipulador, cadeia cinemática.

ABSTRACT

This work deals with the development of a package in ROS (Robot Operating System), for the motion control of a humanoid using teleoperation. We validate this project using inverse kinematics control with dual quaternions. The robotic platform NAO is presented as a robust hardware to the application of control theory used here. To capture human motions, we use the Kinect sensor. The following chapters presents the theory and the development of this initiative.

Keywords: NAO, inverse kinematics, dual quaternions, transformation matrix, control, pseudo inverse, jacobian, control, manipulator, kinematic chain.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos

$\underline{\mathbf{q}}$,$\underline{\mathbf{p}}$	Quatérnio dual
$\mathbf{q}$, $\mathbf{r}$, $\mathbf{p}$, $\mathbf{t}$	Quatérnio unitário
$\underline{a}$,$\underline{b}$	Número dual
$\vec{\theta}$, $\vec{p}$, $\vec{d}$	Vetor
$\vec{i}$, $\vec{j}$, $\vec{k}$, $\vec{i}'$, $\vec{j}'$, $\vec{k}'$	Vetores unitários
t, x, y, z	Escalares
J	Jacobiana
J^T	Transposta da jacobiana
$J^\dagger$	Pseudo-inversa da jacobiana
T	Matriz de transformação homogênea
R	Matriz de rotação

Siglas

ANL	Argonne National Laboratory
ROS	Robotic Operational System
SCARA	Selective Compliance Assembly Robot Arm
DOF	Degrees-Of-Freedom
API	Application Programming Interface
BSD	Berkeley Software Distribution
FSR	Force Sensitive Resistor
DQ	Dual Quaternions
CoM	Center of Mass

Capítulo 1

Introdução

Robôs são estruturas criadas para substituir, ou auxiliar o humano em tarefas pesadas, repetitivas, ou perigosas. O desenvolvimento da robótica permite que eles se insiram com cada vez mais frequência no contexto humano. A utilização dessas máquinas não se limita a arenas em que a presença humana é controlada, ou inexistente. O robô coexiste com o homem em seu ambiente. Mais além disso, é crescente o interesse de que robô execute uma atividade em cooperação com um humano.

Uma máquina que divida espaço com o homem poderia, intuitivamente, apresentar uma morfologia tal que a permita se adaptar facilmente ao ambiente humano, para realizar suas tarefas [1]. Essa é uma das principais motivações para a criação de robôs antropomórficos. Sua estrutura, semelhante à humana facilitaria também a interação homem-máquina.

O estudo de robôs humanóides trouxe uma série de problemas mais complexos que os enfrentados com plataformas móveis com rodas. Um desses desafios está na área da locomoção. Robôs bípedes exigem controle de equilíbrio para que possam se movimentar de forma estável. Além disso, lidamos agora com uma estrutura formada por acoplamentos de manipuladores. Devemos entender também como controlar essas estruturas para executar uma tarefa específica.

É aqui que surge a nossa abordagem do controle de humanóide por teleoperação. A teleoperação pode ser vista como uma transferência de destreza do humano a um outro objeto. Os primeiros tele-manipuladores apareceram cerca dos anos 40, a partir da necessidade de se manipular produtos perigosos, tais como os nucleares [1]. Foi quando Goertz e seu grupo do Laboratório Nacional de Argonne (ANL[1]) desenvolveram uma série de protótipos de manipuladores mecanicamente teleoperados.

A medida que a teleoperação se desenvolvia, os sistemas robóticos foram se tornando mais inteligentes. As primeiras grandes mudanças vieram na introdução de energia. Logo, vieram os avanços na área da computação e da teoria de controle automático que trouxeram mais autonomia às máquinas.

Abstraindo um pouco o conceito de teleoperação, ela pode ser usada ainda para o aprendizado

[1]ANL: acrônimo em inglês de *Argonne National Laboratory*.

de máquinas. Assim, um robô que interage com um humano pode imitar seus movimentos para realizar tarefas complementares.

1.1 Objetivos do projeto

Este trabalho propõe uma solução para o controle de corpo todo de um humanóide, baseado na movimentação humana. Para capturar os movimentos do homem, utilizamos um sensor chamado Kinect. Para o controle, utilizaremos uma abordagem com cinemática inversa[2], a partir das posições dos efetuadores-finais (mãos e pés) do usuário. Utilizaremos duas técnicas para implementar esse controle. Uma delas usando matrizes de transformação homogênea e a outra, usando quatérnios duais. Esses recursos serão detalhados nos próximos capítulos.

1.2 Trabalhos Relacionados

Nesta seção listaremos alguns outros trabalhos que abordaram o tema de teleoperação de humanóides.

Podemos separar esses trabalhos de acordo com o instrumento que foi utilizado para a teleoperação (e.g., joystick, câmera), as técnicas para controle de movimentação (e.g. mapeamento direto, cinemática inversa) e o robô adotado (e.g., NAO). Esse desmembramento nos permitirá diferenciar melhor cada trabalho.

No artigo em [2], o autor escolheu o sistema de captura de movimento, Xsens MVN[3], para teleoperar um NAO. A técnica de controle se baseia no mapeamento direto (angular). No entanto, o robô é treinado para se movimentar corretamente através do uso de redes neurais *feedforward* para cada grau de liberdade.

Em [3], também é usada a vestimenda de sensores inerciais para controlar o NAO. No entanto, as novas posições dos membros do robô são encontradas por cinemática inversa. Devido à diferença na correspondência das juntas do humano e do robô, tomam-se como referência apenas os efetuadores-finais. Neste trabalho, é possível ainda que o NAO movimente suas pernas mantendo a estabilidade.

Zuher, em [4], utiliza o Kinect para capturar dados do humano. É criada uma correspondência entre os ângulos do humano e do robô para implementar um controle baseado em mapeamento direto. Aqui também é utilizada uma técnica para rastrear as posições das juntas do pé e permitir que o NAO se mantenha em equilíbrio quando ele está sobre um pé.

Um outro trabalho interessante foi realizado por Taylor Veltrop, ele utiliza a plataforma ROS para controle de posição dos membros superiores do NAO [4]. Sua implementação utiliza o mapea-

[2]Cinemática inversa: técnica que permite encontrar os dados angulares de uma cadeia cinemática a partir da posição final do efetuador.

[3]Xsens MVN: vestimenta de corpo inteiro equipada com sensores inerciais que permitem a captura de movimento do usuário.

[4]Teleoperação por Taylor Veltrop: mais informações sobre sua implementação podem ser encontradas em http://wiki.ros.org/openni/Contests/ROS%203D/Humanoid%20Teleoperation

mento direto, como [4], mas permite ao usuário utilizar cinemática direta para o robô acompanhar seu movimento.

1.3 Apresentação do manuscrito

O próximo capítulo nos apresenta os conceitos matemáticos necessários para a implementação desse trabalho. Veremos os métodos escolhidos para representação de um corpo rígido, de sua rotação e translação no espaço. Apresentaremos os quatérnios duais, técnica utilizada nesse trabalho para controle de movimentação do robô. O Capítulo 3 apresenta as ferramentas de software utilizadas e as técnicas para utilizarmos os conceitos do Capítulo 2 em um humanóide NAO. Em seguida, o Capítulo 4 expõe os resultados desse trabalho. Será possível verificar q eficiência da utilização dos quatérnios duais no controle do autômato. Finalmente, o Capítulo 5 apresenta as últimas considerações do trabalho.

Capítulo 2

Fundamentação

Esta capítulo busca dar ao leitor a base conceitual e matemática necessária para que ele compreenda os principais métodos usados na modelagem do robô para seu controle cinemático.

2.1 Introdução

O controle de trajetória do corpo de um humanóide é uma tarefa complexa. Para melhor analisar esse tipo de sistema, devemos dividí-lo em partes menores.

Para isso, separamos o robô em cadeias cinemáticas. Essa simplificação nos permitirá utilizar a metodologia existente para modelagem e controle de manipuladores, no controle do NAO. O robô humanóide estudado neste trabalho é constituido por 5 cadeias cinemáticas: cabeça, dois membros superiores e dois membros inferiores.

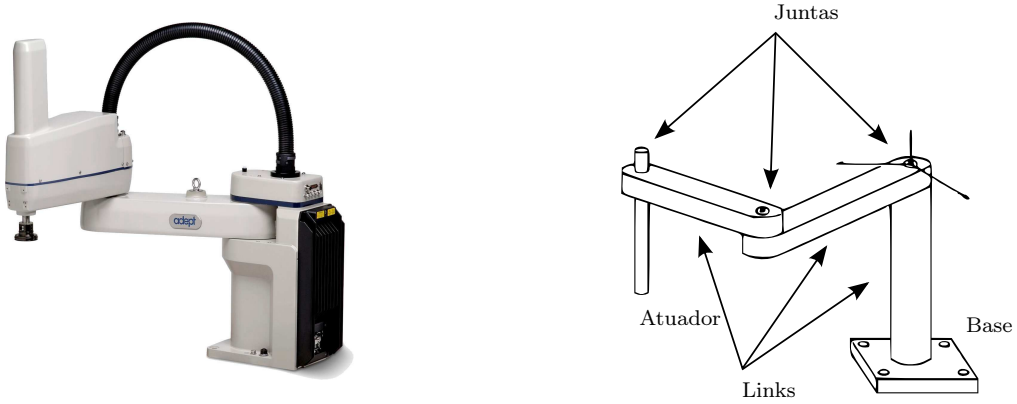


Figura 2.1: Braço robótico SCARA (Selective Compliance Assembly Robot Arm) desenvolvido na Universidade de Yamanashi. À esquerda temos o modelo real do braço (fonte: Direct Industry) e à direita seu esquemático (fonte: Wikipédia).

As cadeias cinemáticas são um conjunto de corpos rígidos (referenciados como *links*) conectados por juntas. Um dos limites da cadeia é chamado de base, enquanto o outro é chamado de efetuador-

final. A Figura 2.1 apresenta as partes da cadeia cinemática, i.e. efetuador-final, *links*, juntas e base.

Para que seja possível implementar um algoritmo de controle no robô, é necessário que tenhamos a sua modelagem matemática. Neste capítulo, estaremos preocupados em apresentar a base matemática para o desenvolvimento e a manuseio de métodos matemáticos para manipuladores.

Inicialmente, a Seção 2.2 apresenta como descrever um corpo no espaço tridimensional. Ela nos introduz técnicas para aplicar transformações de translação e rotação no mesmo (matrizes de transformação, quatérnios e quatérnios duais). Em seguida, a Seção 2.3 aborda a análise cinemática de uma cadeia de corpos rígidos ligados por juntas. Finalmente, a Seção 2.4 aborda alguns métodos que tornam possível o controle de movimentação do humanóide.

2.2 Representação e Movimentação de Corpos Rígidos

A representação de um corpo rígido no espaço é dada pela sua posição e orientação em relação a um sistema de referência. Na Figura 2.2, temos o sistema de coordenadas base com três eixos ortonormais, $(x,\ y,\ z)$, os vetores unitários correspondentes a cada eixo $(\vec{i},\ \vec{j},\ \vec{k})$ e um corpo rígido com centro de massa no ponto P.

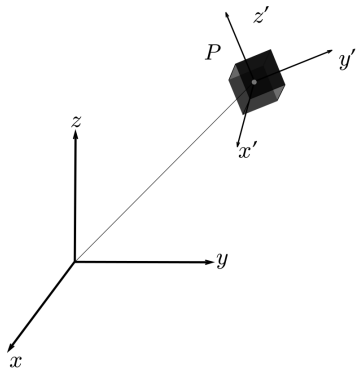


Figura 2.2: Representação de um corpo rígido no espaço

Esse objeto possui um sistema de coordenadas ortonormal próprio $(x',\ y',\ z')$, acoplado ao mesmo, com vetores unitários $(\vec{i}',\ \vec{j}',\ \vec{k}')$. A origem desse sistema localiza-se em um ponto de interesse do corpo, normalmente no centro de massa, ou centróide, como é o caso na figura exemplo.

A movimentação de uma partícula no espaço Euclidiano é dada pela variação da localização da mesma no tempo tendo-se sempre como referência um *frame* de coordenadas inercial [5]. Utilizaremos ferramentas de álgebra linear para descrever o movimento de um corpo rígido no espaço.

Quando o corpo rígido muda de posição, ou de orientação, todas as distâncias entre seus pontos devem ser preservadas. Assim, um modelo matemático que descreva uma movimentação deve guardar essa propriedade da colinearidade, além de outras mencionadas a seguir. Uma transformação de um corpo rígido que faz um mapeamento $f : \mathbb{R}^n \to \mathbb{R}^m$ deve satisfazer as seguintes propriedades [5]:

- A distância entre os pontos é preservada: $\| f(p) - f(q) \| = \| p - q \|$, para todos os pontos p, $q \in \mathbb{R}^n$; e
- O produto vetorial é preservado: $f * (v \times w) = f * (v) \times f * (w)$, para todos os vetores v, $w \in \mathbb{R}^n$.

A posição e a orientação de um robô podem ser parametrizados de diferentes formas, tais como:

- Matrizes de rotação e quatérnios unitários (rotação);
- Matrizes de translação (posição);
- Matrizes homogêneas e quatérnios duais (posição e orientação).

Falaremos brevemente sobre cada um desses métodos.

A posição linear do objeto da Figura 2.2 pode ser representada por um vetor com origem no *frame* de referência e fim na origem do *frame* do objeto:

$$\vec{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \in \mathbb{R}^3.$$

Os componentes desse vetor consistem na projeção do ponto de interesse em cada um dos eixos do sistema de coordenadas de referência.

A orientação de um corpo rígido pode ser descrita por diferentes métodos numéricos. Alguns deles são: ângulos de Euler[1] (Figura 2.3), matriz de cossenos diretores [2] (Figura 2.3) e quatérnios. Usaremos alguns deles neste trabalho.

Para representar a orientação de um objeto, é conveniente considerar o sistema de coordenadas fixo ao mesmo. Chamaremos esse *frame* de local. Os vetores unitários do *frame* local são representados com respeito ao *frame* de referência. Esse *frame* local acompanha o corpo em todos os seus movimentos.

Podemos expressar a orientação do corpo em função do sistema de coordenadas de referência pelas seguintes equações:Esses três vetores podem ser combinados juntos em uma matriz de 3 dimensões, R:

$$R = \begin{bmatrix} \vec{i'} & \vec{j'} & \vec{k'} \end{bmatrix}.$$

Os componentes desses vetores correspondem às rotações relativas a cada eixo do *frame* de referência inerte. A matriz de coeficientes do vetor $\vec{i'}$, $\vec{j'}$, $\vec{k'}$ é chamada de matriz de rotação.

[1] Ângulos de Euler: representam a orientação de um corpo rígido em relação a cada eixo do sistema de coordenadas tridimensional. Esses ângulos recebem os nomes de ângulo de guinada (em inglês, *yaw*), ângulo de arfagem (em inglês, *pitch*) e ângulo de rolagem (em inglês, *roll*).

[2] Matriz de cossenos diretores: matriz formada a partir de todas as combinações possíveis entre os vetores unitários do corpo e do sistema de referência.

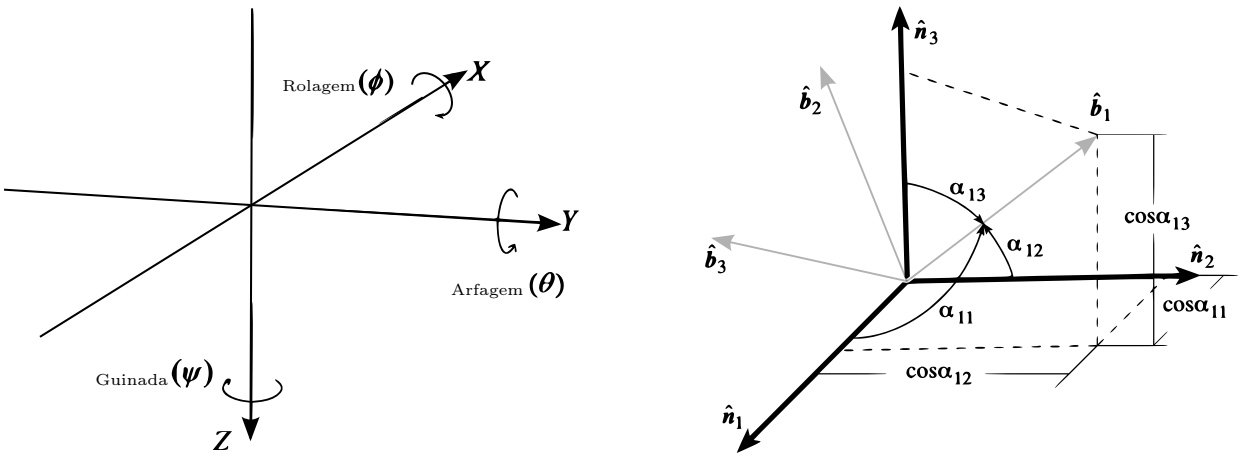


Figura 2.3: Representação espacial da orientação de um corpo rígido em três dimensões. À esquerda, temos os ângulos de Euler e à direita, os cossenos diretores (fonte: VirginiaTech).

2.2.1 Matrizes de Transformação Homogênea

A movimentação de um corpo rígido no espaço pode ser modelada como uma transformação linear representada por matrizes de transformação.

Os tipos de transformação linear mais comuns são os de rotação, de escala, de reflexão, de translação e de projeção ortogonal.

A matriz afim consiste na combinação de um, ou de mais tipos de transformações lineares. Buscamos então uma representação para a translação e rotação de um objeto, já que esses são os tipos de movimentação possíveis em nosso robô.

Uma transformação linear $f : \mathbb{R}^n \to \mathbb{R}^m$, entre dois espaços afins, possui a forma $f(\vec{p}) = R\vec{p}+\vec{d}$, onde R é uma matriz $(n \times n)$ que representa uma transformação linear (apenas rotação neste trabalho), $\vec{d}$, um vetor coluna de dimensão n que representa uma translação e $\vec{p}$ o vetor que representa o ponto no qual é aplicada a transformação.

O mesmo modelo pode ser representado pela matriz aumentada de transformação T a seguir. Para nosso sistema tridimensional, teremos a seguinte matriz de transformação afim:

$$T = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & d_x \\ R_{yx} & R_{yy} & R_{yz} & d_y \\ R_{xx} & R_{yz} & R_{zz} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Aplicando uma rotação definida pela matriz R e uma translação $\vec{d}$ a um vetor coluna $\vec{p} = (p_x, p_y\, p_z)^T$, encontramos o vetor resultante $\vec{p'}$:

$$\vec{p'} = T\vec{p} \Longrightarrow \begin{bmatrix} \vec{v'_x} \\ \vec{v'_y} \\ \vec{v'_z} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & d_x \\ R_{yx} & R_{yy} & R_{yz} & d_y \\ R_{xx} & R_{yz} & R_{zz} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{p_x} \\ \vec{p_y} \\ \vec{p_z} \\ 1 \end{bmatrix}.$$

Transformações consecutivas sobre um corpo são descritas por multiplicações de matrizes afins. Seja T_i^j a matriz referente a uma transformação entre espaços afins do *frame* i em relação ao *frame* j, temos:

$$\vec{p'} = T_2^0\vec{p} = T_1^0T_2^1\vec{p}. \text{ Então,}$$

$$\vec{p'} = \begin{bmatrix} R_1^0 & \vec{d_1^0} \\ 0...0 & 1 \end{bmatrix} \begin{bmatrix} R_2^1 & \vec{d_2^1} \\ 0...0 & 1 \end{bmatrix} \begin{bmatrix} \vec{p} \\ 1 \end{bmatrix} = \begin{bmatrix} R_1^0R_2^1 & R_1^0\vec{d_2^1} + \vec{d_1^0} \\ 0...0 & 1 \end{bmatrix} \begin{bmatrix} \vec{p} \\ 1 \end{bmatrix}.$$

Aqui o vetor $\vec{p}$ sofre duas transformações T_1^0 e T_2^1 consecutivas. Essa operação também é chamada de composição de transformações.

Rotação com matrizes

Uma rotação no plano cartesiano pode ser descrita por uma função que rotaciona vetores em um ângulo fixo e em uma direção específica. As matrizes de rotação são usadas para descrever a mudança de orientação de um objeto rígido em torno de um ponto de referência.

Uma rotação em um espaço tridimensional é descrita como uma matriz R de dimensão 3×3 de determinante unitário. É importante destacar que os vetores-coluna da matriz R são mutualmente ortogonais, já que representam o coeficientes dos vetores unitários de um *frame* ortonormal, ou seja,

$$R^T = R^{-1};\ RR^T = R^TR = I;\ det(R) = 1.$$

A matriz R pode ser representada genericamente da seguinte forma:

$$R = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{xx} & R_{yz} & R_{zz} \end{bmatrix}.$$

Um método para descrição da orientação de uma *frame* de coordenada em relação a outro pode ser dado usando-se os ângulos α, β, γ. Eles correspondem à rotação em torno de x ($R_x(\alpha)$), y ($R_y(\beta)$) e z ($R_z(\gamma)$), respectivamente (ângulos de Euler). As rotações são positivas caso realizadas no sentido horário, em relação a cada um dos eixos. Temos:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix},$$

$$R_y(\beta) = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix},$$

$$R_z(\gamma) = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Lembrando que os elementos de cada vetor coluna descrevem a orientação dos vetores unitários do *frame* local em referência ao *frame* mundo.

Para representar duas, ou mais rotações consecutivas, é possivel usar a composição de matrizes. Assim, uma rotação em x, seguida de uma rotação em z e uma em z sobre um vetor $\vec{v}$ é descrita da seguinte forma:

$$\vec{p'} = R_z(\gamma)R_y(\beta)R_x(\alpha)\vec{p}.$$

Operações de rotação não comutam, ou seja a ordem de aplicação das rotações é importante. Então:

$$R_y(\beta)R_x(\alpha) \neq R_x(\alpha)R_y(\beta).$$

Finalmente, podemos representar uma transformação pura com a matriz afim expandida (eq. 2.1) fazendo $d_x = d_y = d_z = 0$:

$$T_{\text{rotação pura}} = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & 0 \\ R_{yx} & R_{yy} & R_{yz} & 0 \\ R_{xx} & R_{yz} & R_{zz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

Translação com matrizes

A translação é uma transformação geométrica que corresponde à idéia de deslocamento de um objeto.

Podemos representar uma translação usando a matriz afim expandida da seguinte forma:

$$T_{\text{translação pura}} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

Perceba que a matriz de rotação aqui é substituída por uma matriz identidade de tamanho correspondente à dimensão do nosso sistema. Os componentes d_x, d_y e d_z correspondem aos deslocamentos referentes aos eixos x, y e z.

É fácil verificar que aplicando-se dois deslocamentos consecutivos, o resultado da translação é a soma desses desvios em relação a cada eixo. Sendo I_{3x3} a matriz identidade de dimensão 3, temos:

$$T_3^1 = T_2^1 T_3^2 = \begin{bmatrix} & & d^1_{x2} \\ & I_{3x3} & d^1_{y2} \\ & & d^1_{z2} \\ 0...0 & & 1 \end{bmatrix} \begin{bmatrix} & & d^2_{x3} \\ & I_{3x3} & d^2_{y3} \\ & & d^2_{z3} \\ 0...0 & & 1 \end{bmatrix} = \begin{bmatrix} & & d^1_{x2} + d^2_{x3} \\ & I_{3x3} & d^1_{y2} + d^2_{y3} \\ & & d^1_{z2} + d^2_{z3} \\ 0...0 & & 1 \end{bmatrix}.$$

2.2.2 Quatérnios Unitários

Os quatérnios são uma extensão dos números complexos para $\mathbb{R}^4$. Sua invenção é atribuída a W. R. Hamilton em 1843 e formalizada apenas em 1866 com a publicação de [6].

De forma análoga aos números complexos, que podem descrever uma rotação em um plano, os quatérnios são uma estrutura algébrica que descreve rotações no espaço. Comparando-se com outras modelagens, como por exemplo as matrizes de rotação e os ângulos de Euler, os quatérnios são mais estáveis numericamente e podem ser mais eficientes. Além disso, os quatérnios permitem nos aproximar de soluções de sistemas algébricos integrando parâmetros de rotação desconhecidos. Eles possuem diversas aplicações nos campos da robótica, dos video games, da cinemática, da física, do processamento de sinal, etc.

Vimos na subseção anterior que uma rotação em $\mathbb{R}^3$ poderia ser representada por uma matriz ortogonal de 3 dimensões de determinante unitário. Essa representação, portanto, aparenta redundância, já que apenas 3 elements dentre os 9 são independentes.

Como veremos a seguir, os quatérnios são uma estrutura mais simples e mais facilmente compreensível geometricamente que as matrizes de rotação. Eles são formados por 4 elementos que compõem uma 4-tupla.

Os quatérnios são gerados juntando-se as unidades imaginárias i, j e k a números reais. Podemos representar um quatérnio como o seguinte conjunto:

$$\mathbb{H} = \{t + x\vec{i} + y\vec{j} + z\vec{k};\ t,\ x,\ y,\ z \in \mathbb{R}\}. \tag{2.4}$$

Os vetores unitários $\vec{i} = [1,\ 0,\ 0]^T$, $\vec{j} = [0,\ 1,\ 0]^T$ e $\vec{k} = [0,\ 0,\ 1]^T$ possuem a seguinte característica:

$$i^2 = j^2 = k^2 = -1 \tag{2.5}$$

e se relacionam da seguinte forma:

$$\begin{array}{ll} \vec{i}\vec{j} = \vec{k}; & \vec{j}\vec{i} = -\vec{k}; \\ \vec{j}\vec{k} = \vec{i}; & \vec{k}\vec{j} = -\vec{i}; \\ \vec{k}\vec{i} = \vec{j}; & \vec{i}\vec{k} = -\vec{j}. \end{array} \tag{2.6}$$

Dado um certo quatérnio, $\mathbf{q} = t + x\vec{i} + y\vec{j} + z\vec{k}$, podemos reescrevê-lo de forma tal que $\mathbf{q} = (t, \vec{v}) = [t,\ x,\ y,\ z]^T$.

Operações elementares com quatérnios

Dados $\mathbf{q}_1 = (t_1, \vec{v_1})$ e $\mathbf{q}_2 = (t_2, \vec{v_2})$, apresentamos na Tabela 2.1 as operações aritméticas elementares dos quatérnios.

Na definição de multiplicação na Tabela 2.1 utilizamos "." para representar um produto interno e "$\times$", ou nenhum símbolo entre os componentes, para representar um produto externo.

Operação Aritmética	Descrição
Multiplicação por um escalar	$\lambda\mathbf{q} = (\lambda t,\ \lambda\vec{v})$
Adição	$\mathbf{q_1} + \mathbf{q_2} = (t_1 + t_2, \vec{v_1} + \vec{v_2})$
Multiplicação (produto externo)	$\mathbf{q_1 q_2} = (t_1 t_2 - \vec{v_1}.\vec{v_2},\ t_1\vec{v_2} + t_2\vec{v_1} + \vec{v_1} \times \vec{v_2})$
Conjugada	$\mathbf{q}* = (t, -\vec{v})$
Norma (ou magnitude)	$\parallel \mathbf{q} \parallel^2 = \mathbf{q}\mathbf{q}*$
Inversa	$\mathbf{q}^{-1} = {\mathbf{q}*}/{\parallel\mathbf{q}\parallel^2}$

Tabela 2.1: Operações aritméticas elementáres para quatérnios

Pela definição da inversa de um quatérnio $\mathbf{q}$, verificamos facilmente que:

$$\mathbf{q}^{-1}\mathbf{q} = \mathbf{q}\mathbf{q}^{-1} = 1.$$

Caso $\mathbf{q}$ seja um quatérnio unitário, temos:

$$\mathbf{q}^{-1} = \mathbf{q}^*. \tag{2.7}$$

Rotações com quatérnios

Os quatérnions unitários são aqueles onde $\parallel \mathbf{q} \parallel = 1$. Eles são usados para representar uma rotação, ou uma orientação, podendo ser chamados de quatérnios de rotação, ou de orientação, respectivamente.

Consideremos o vetor $\vec{v}$, descrito anteriormente, onde $\vec{i}$, $\vec{j}$ e $\vec{k}$ são vetores unitários representando os três eixos cartesianos. Uma rotação de um ângulo θ em torno de um eixo definido por esse vetor $\vec{v}$ é representada por um quatérnio usando uma extensão da fórmula de Euler:

$$\mathbf{q} = e^{\frac{1}{2}\theta(x\vec{i}+y\vec{j}+z\vec{k})} = cos\frac{1}{2}\theta + (x\vec{i} + y\vec{j} + z\vec{k})sin\frac{1}{2}\theta. \tag{2.8}$$

Simplificando,

$$\mathbf{q} = (cos\frac{1}{2}\theta, \vec{v}sin\frac{1}{2}\theta). \tag{2.9}$$

Sendo $\mathbf{q}$ um quatérnio unitário, temos que $t^2 + \|\vec{v}\|^2 = 1$. Isso implica que sempre existe um ângulo θ que satisfaça (2.10) e (2.11).

$$cos^2\theta = t^2, \tag{2.10}$$

$$sin^2\theta = \|\vec{v}\|^2. \tag{2.11}$$

Com isso podemos validar (2.8) e (2.9).

Tomemos o vetor $\vec{p}$ já usado anteriormente para aplicarmos transformações matriciais, com o intuito agora de executarmos uma rotação representada pelo quatérnio unitário $\mathbf{q}$. Esse vetor pode ser reescrito como um quatérnio puro, já que sua parte real é nula, $\mathbf{p} = (0, \vec{p})$.

O resultado dessa operação é expresso então por:

$$\mathbf{r}_{rot}(\mathbf{p}) = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}.$$

Sendo $\mathbf{q}$ um quatérnio unitário, segundo (2.7), podemos reescrever a fórmula como:

$$\mathbf{r}_{rot}(\mathbf{p}) = \mathbf{q}\mathbf{p}\mathbf{q}^{*}.$$

Dessa operação, teremos a parte real nula e a parte imaginária conterá o resultado da rotação de θ rad em torno do eixo definido por $\vec{v}$.

Caso desejemos aplicar duas, ou mais rotações em um corpo, podemos usar a composição de quatérnios. Para duas rotações distintas representadas por $\mathbf{q}_1^0$ e $\mathbf{q}_2^1$ sobre um ponto $\mathbf{p}$, temos:

$$\mathbf{q}_2^1(\mathbf{q}_1^0(\mathbf{p})) = \mathbf{q}_2^1(\mathbf{q}_1^0\mathbf{p}\mathbf{q}_1^{0*}) = \mathbf{q}_2^1\mathbf{q}_1^0\mathbf{p}\mathbf{q}_1^{0*}\mathbf{q}_2^{1*}.$$

Fazendo,

$$\mathbf{q}_2^0 = \mathbf{q}_2^1\mathbf{q}_1^0,$$
$$\mathbf{q}_2^{0*} = (\mathbf{q}_2^1\mathbf{q}_1^0)^* = \mathbf{q}_1^{0*}\mathbf{q}_2^{1*}.$$

Então, podemos representar a composição das rotações $\mathbf{q}_1^0$ e $\mathbf{q}_2^1$ diretamente por $\mathbf{q}_2^0$.

2.2.3 Quatérnios Duais

Os quatérnios duais são uma ferramenta matématica conscisa e eficiente para representar rotação e translação simultaneamente [7]. Eles apresentam uma estrutura unificada de 8 componentes combinando as duas transformações.

Algoritmos baseados em quatérnios duais são mais eficientes que os que se baseiam na representação matricial. Eles têm melhores propriedades como velocidade constante, caminho mais curto, invariância de coordenadas. Algumas outras vantagens incluem serem facilmente convertidos em outras formas (e.g. matrizes) e a sua não ambiguidade.

Antes de falarmos sobre os quatérnios duais, será feita uma breve introdução aos números duais.

Números duais

Os números duais são uma extensão dos números reais pelo uso do elemento ϵ, o fator dual, com a propriedade $\epsilon^2 = 0$, sendo $\epsilon \neq 0$. Clifford [8] introduziu sua álgebra geométrica como uma generalização da álgebra de Grassman, dos números complexos e dos quatérnios [9].

Todo número dual possui a forma:

$$\underline{z} = a + \epsilon b. \tag{2.12}$$

Sendo a chamado de a parte real, b de parte dual e finalmente. Usualmente, a e b são elementos do mesmo tipo (escalares, vetores, quatérnios, etc).

Operação Aritmética	Descrição
Multiplicação por um escalar	$\lambda\underline{\mathbf{q}} = \lambda\underline{\mathbf{q}}_p + \epsilon\lambda\underline{\mathbf{q}}_d$
Adição	$\underline{\mathbf{q}}_1 + \underline{\mathbf{q}}_2 = \underline{\mathbf{q}}_{p1} + \underline{\mathbf{q}}_{p2} + \epsilon(\underline{\mathbf{q}}_{d1} + \underline{\mathbf{q}}_{d2})$
Multiplicação (produto externo)	$\underline{\mathbf{q}}_1\underline{\mathbf{q}}_2 = (\underline{\mathbf{q}}_{p1}\underline{\mathbf{q}}_{p2}) + \epsilon(\underline{\mathbf{q}}_{p1}\underline{\mathbf{q}}_{d2} + \underline{\mathbf{q}}_{d1}\underline{\mathbf{q}}_{p2})$
Conjugada	$\underline{\mathbf{q}}* = \underline{\mathbf{q}}_p * + \epsilon\underline{\mathbf{q}}_d*$
Norma (ou magnitude)	$\parallel \underline{\mathbf{q}} \parallel^2 = \underline{\mathbf{q}}\underline{\mathbf{q}}*$
Inversa	$\underline{\mathbf{q}}^{-1} = \frac{\underline{\mathbf{q}}*}{\parallel\underline{\mathbf{q}}\parallel^2}$

Tabela 2.2: Operações aritméticas elementáres para quatérnios duais

O fator dual é usado para distinguir as partes real e dual de forma análoga ao que ocorre com os números complexos. Neles, utilizamos o número complexo $\vec{i}$ para separar as partes real e imaginária. A comparação porém é muito mais complexa que isso. Por isso nos ateremos a sua definição em (2.12).

(escrever sobre a inversa?)

Características dos quatérnios duais

Eles se utilizam da algebra de Clifford e dos quatérnios para representar translações e rotações de um corpo rígido.

Um quatérnio dual $\underline{\mathbf{q}}$ pode ser definido por:

$$\underline{\mathbf{q}} = \mathbf{q}_p + \epsilon\mathbf{q}_d. \tag{2.13}$$

Como as partes real e dual são quatérnios, podemos expandir (2.13) em:

$$\underline{\mathbf{q}} = t_p + x_p\vec{i} + y_p\vec{j} + z_p\vec{k} + \epsilon(t_d + x_d\vec{i} + y_d\vec{j} + z_d\vec{k}). \tag{2.14}$$

Ou, podemos representar um dual quatérnio como um par ordenado de quatérnios:

$$\underline{\mathbf{q}} = (\mathbf{q}_p,\, \mathbf{q}_d).$$

Operações elementares com quatérnios duais

As operações com quatérnios se assemelham às dos quatérnios [10]. As operações mais elementares estão listadas na Tabela 2.2.

Rotação e translação com quatérnios duais:

Como vimos na Subseção anterior, uma rotação de θ rad em torno do eixo definido por $\vec{n}$, pode ser representada pelo quatérnio:

$$\mathbf{r} = (cos\frac{1}{2}\theta, \vec{n}sin\frac{1}{2}\theta).$$

Sendo,

$$\vec{n} = n_x\vec{i} + n_y\vec{j} + n_z\vec{k},$$

e uma translação pode ser representada pelo quatérnio puro:

$$\mathbf{t} = (t_x\vec{i} + t_y\vec{j} + t_z\vec{k}).$$

Assim, um quatérnio dual unitário, tal que $\left\|\underline{\mathbf{q}}\right\| = 1$, representa a translação $\mathbf{t}$ e a rotação $\mathbf{r}$ de um corpo rígido de acordo com:

$$\underline{\mathbf{q}} = \mathbf{r} + \frac{1}{2}\epsilon\mathbf{t}\mathbf{r}. \tag{2.15}$$

A partir de (2.15), podemos encontrar a representação de uma rotação pura fazendo $\mathbf{t} = [0, 0, 0, 0]$, de forma semelhando ao realizado com matrizes em (2.2). Essa operação zera a parte dual de $\underline{\mathbf{q}}$, resultando em:

$$\underline{\mathbf{q}}_{\text{rotação pura}} = (\mathbf{r}, 0) = [cos\frac{1}{2}\theta,\ n_x sin\frac{1}{2}\theta,\ n_y sin\frac{1}{2}\theta,\ n_z sin\frac{1}{2}\theta,\ 0,\ 0,\ 0\,0]^T.$$

Para representar uma translação pura, seguimos um raciocínio semelhante ao realizado com a matriz de translação em (2.3). Estabelecemos $\mathbf{r} = [1, 0, 0, 0]$, então:

$$\underline{\mathbf{q}}_{\text{translação pura}} = [1, 0, 0, 0, 0, t_x, t_y, t_z]^T. \tag{2.16}$$

Com essas ferramentas, podemos agora aplicar uma transformação de translação e rotação simultâneas sobre um ponto, definido pelo vetor $\vec{p}$, usando quatérnio dual unitário definido em (2.15). Essa transformação ocorre de forma semelhante à transformação de rotação com quatérnios.

Inicialmente $\vec{p}$ deve ser reescrito na forma de quatérnio dual, como em (2.16). Então,

$$\underline{\mathbf{p}}' = \underline{\mathbf{q}\mathbf{p}\mathbf{q}}^*.$$

A composição de quatérnios duais também ocorre de forma semelhante à dos quatérnios. Para duas transformações consecutivas, teríamos:

$$\underline{\mathbf{q}}_1^0(\underline{\mathbf{q}}_2^1(\underline{\mathbf{p}})) = \underline{\mathbf{q}}_2^0\underline{\mathbf{p}}\underline{\mathbf{q}}_2^{0*}.$$

Tal que,

$$\underline{\mathbf{q}}_2^0 = \underline{\mathbf{q}}_1^0\underline{\mathbf{q}}_2^1,\ e\ \underline{\mathbf{q}}_2^{0*} = \underline{\mathbf{q}}_2^{1*}\underline{\mathbf{q}}_1^{0*}.$$

2.3 Cinemática do Robô

A análise cinemática da estrutura mecânica do robô consiste na descrição da movimentação do mesmo em relação a um *frame* de referência físico [11]. Ignoram-se as forças e os momentos que causaram a movimentação da estrutura.

2.3.1 Acoplamento de Corpos Rígidos

Corpos rígidos podem ser acoplados entre si através das juntas. Diferentes tipos de acoplamento permitem diversas formas de movimentação. Podemos classificar os acoplamentos mecânicos em dois tipos: de alta e de baixa ordem. Os pares de baixa ordem têm uma superfície, ou um plano de contato, como as juntas prismáticas e de revolução Figura 2.4. Os de alta ordem possuem uma linha, ou um ponto de contato, como uma engrenagem[3].

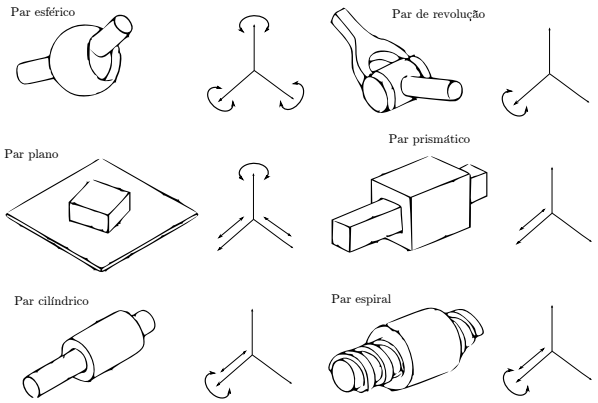


Figura 2.4: Tipos de acoplamentos mecânicos de baixa ordem (fonte: MIT).

Sucessivos acoplamentos de corpos por juntas formam as cadeias cinemáticas. Como o robô com o que trabalhamos possui apenas juntas de rotação, podemos assumir que cada acoplamento possui apenas 1-DOF[4] descrito pelo valor da sua rotação. Cada uma das cadeias do robô é constituída de i juntas e $i+1$ *links*, já que uma junta conecta dois *links*. Assim, numeraremos as juntas de 1 a i e os links de 0 a i.

2.3.2 Cinemática Direta

A problemática da cinemática direta diz respeito à relação entre os valores da configuração de cada junta (rotação) e a posição e orientação do atuador final [11]. Ou seja, ela define um mapeamento do espaço de juntas ao espaço tridimensional.

A equação cinemática que descreve a relação entre a posição da ponta atuadora da cadeia cinemática e os valores das juntas da mesma é dada a seguir:

$$\vec{x} = f(\vec{\theta}), \tag{2.17}$$

sendo $\vec{x} \in \mathbb{R}^m$ e $\vec{\theta} \in \mathbb{R}^n$, então $f : \mathbb{R}^n \to \mathbb{R}^m$.

Para encontrarmos a equação cinemática de uma cadeia, usamos transformações rígidas que descrevam a rotação, ou translação de uma junta em relação à junta anterior.

Para padronizar os *frames* de coordenadas dos links de uma cadeia cinemática, Jaqcues Denavit e Richard Hartenberg apresentaram em 1955 uma convenção que é hoje padrão na robótica [12].

[3]Fonte: http://web.mit.edu/mecheng/pml/spec_config.htm

[4]DOF: Abreviação da expressão em inglês *Degrees-Of-Freedom* (graus de liberdade).

Parâmetro	Descrição
a_i	Distância entre O_i e $O_{i'}$.
d_i	Distância entre O_{i-1}e O_i ao longo do eixo z_{i-1}.
α_i	Ângulo de rotação entre z_{i-1}e z_i em torno de x_i.
θ_i	Ângulo de rotação entre x_{i-1}e x_i em torno de z_i.

[1] *Seu valor é positivo caso a rotação seja feita no sentido anti-horário.*

Tabela 2.3: Parâmetros de Denavit-Hartenberg

No entanto, foi Richard Paul quem, em 1981, demostrou o uso desse sistema para uso em cadeias cinemáticas robóticas [13].

A seguir, veremos como utilizar a convenção de Denavit-Hartenberg para encontrarmos $f(\vec{\theta})$.

2.3.2.1 A Convenção de Denavit-Hartenberg

A notação de Denavit-Hartenberg nos permite caracterizar a posição relativa entre dois sólidos com apenas 4 parâmetros. Através dela podemos encontrar a equação de cinemática direta para uma cadeia aberta de um manipulador através de uma expressão recursiva.

O objetivo é achar a transformada de uma junta i para uma junta $i+1$. Para isso, cada junta deve corresponder a um *frame* de coordenadas próprio de três eixos ortogonais entre si. Busca-se então estabelecer a relação entre esses eixos de acordo com sua posição e orientação.

Convenção padrão Para isso, devemos seguir os passos segundo [11]:

1. Escolher o eixo z_i de acordo com o eixo de movimentação (rotacional, ou translacional) da junta $i+1$;
2. Localizar a origem O_i na interseção de z_i e z_{i-1}, ou na interseção da normal comum[5] entre o eixo z_i e z_{i-1} e o eixo z_i;
3. Estabelecer o eixo $x_i$de acordo com 2.18, ou sobre a normal comum entre z_i e z_{i-1}quando esses eixos forem paralelos; e
4. Finalmente, y_i pode ser encontrado usando-se a regra da mão direita de z_i para x_i.

$$x_i = \pm \frac{(z_{i+1} \times z_i)}{\|z_{i+1} \times z_i\|}. \tag{2.18}$$

Estabelecidos os links, especificamos a posição e a orientação de um *frame* i em relação ao seu *frame* anterior $i-1$ de acordo com os parâmetros da Tabela.

Tendo-se esses parâmetros, podemos encontrar a transformação de um *frame* i em relação ao *frame* $i-1$. Cada parâmetro corresponde a uma transformada de translação, ou de rotação. Para o link i, temos:

[5] A normal comum entre duas linhas é o segmento que caracteriza a distância mínima entre elas.

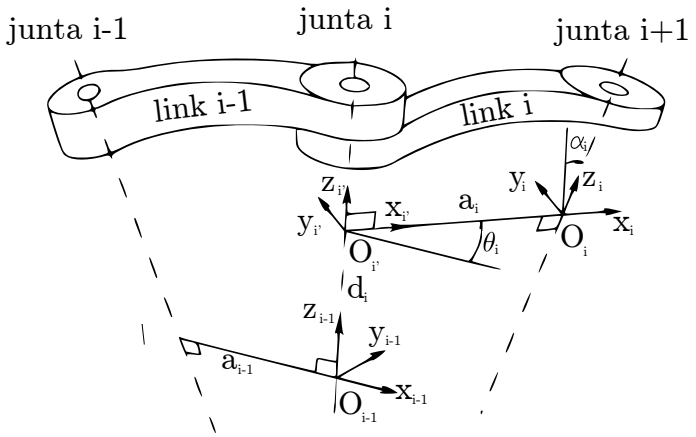


Figura 2.5: Parâmetros da notação clássica de Denavit-Hartenberg de uma cadeia cinemática (fonte: [11]).

Em representação matricial, sendo $T_{rot,a}$ uma matriz de rotação em torno do eixo a e $T_{trans,a}$ uma matriz de translação ao longo do eixo a:

$$T_i^{i-1} = T_{DH} = T_{rot,z}(\theta)T_{trans,z}(d)T_{trans,x}(a)A_{rot,x}(\alpha). \tag{2.19}$$

Em dual-quatérnios:

$$\underline{\mathbf{q}}_i^{i-1} = \underline{\mathbf{q}}_{DH} = \underline{\mathbf{q}}_{trans,z}(\theta)\underline{\mathbf{q}}_{trans,z}(d)\underline{\mathbf{q}}_{trans,x}(a)\underline{\mathbf{q}}_{rot,x}(\alpha). \tag{2.20}$$

Para uma cadeia cinemática de i juntas, a transformação completa da base ao atuador é encontrara através de multiplicações sucessivas.

$$T_i^0 = T_1^0 T_2^1 ... T_i^{i-1}. \tag{2.21}$$

Podemos calcular a equação de cinemática direta com dual-quatérnios seguindo o mesmo princípio:

$$\underline{\mathbf{q}}_i^0 = \underline{\mathbf{q}}_1^0 \underline{\mathbf{q}}_2^1 ... \underline{\mathbf{q}}_i^{i-1}. \tag{2.22}$$

Convenção modificada

A diferença entre a notação clássica de Denavit-Hartenberg e a modificada é a localização dos sistemas de coordenadas de cada link. Basicamente o que acontece é que as coordenadas do *frame* O_{i-1} estão no link $i-1$ e não em i, como na convenção clássica. A mesma idéia segue para as outras juntas.

Na Figura 2.6 podemos ter uma percepção visual da convenção modificada.

Utilizaremos essa convenção no nosso trabalho.

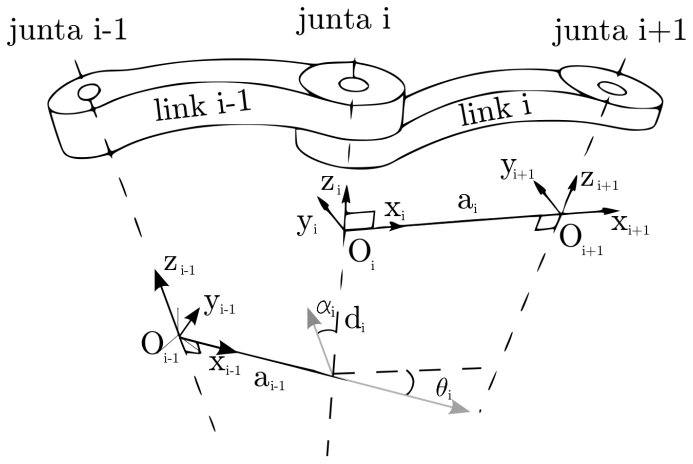


Figura 2.6: Parâmetros da notação modificada de Denavit-Hartenberg

2.3.3 Cinemática Inversa

A cinemática inversa envolve o problema contrário ao da cinemática direta, como já era de se esperar pelo nome. Trata-se da determinação das variáveis das juntas (ângulo de cada DOF) correspondente a uma posição e orientação do efetuador-final.

Podemos encontrar a equação da cinemática inversa analiticamente invertendo a equação de cinemática direta encontrada com a notação de Denavit-Hartenberg.

$$\vec{\theta} = f^{-1}(\vec{x}). \tag{2.23}$$

A multiplicidade de soluções faz da cinemática inversa uma relação e não um mapeamento. Segundo [11], alguns outros problemas que tornam a cinemática inversa mais complexa que a direta incluem:

- As equações são geralmente não lineares. Podendo, ou não haver solução de forma fechada;
- No caso de redundância de juntas, pode haver infinitas soluções;
- Pode haver ainda soluções não-admissíveis fisicamente, de acordo com a estrutura do manipulador.

2.3.4 Cinemática Diferencial

Vimos até agora como estabelecer a relação entre os valores das juntas de uma cadeia cinemática e a posição do atuador final do mesmo em relação à base (cinemática direta) e vice-versa (cinemática inversa). O objetivo desta subseção agora é o de apresentar a relação entre as velocidades das juntas e o seus efeitos sobre as velocidades linear e angular do atuador final. A cinemática diferencial é determinada pela Jacobiana da função de cinemática direta.

Cinemática Direta

A Jacobiana é a matriz que relaciona as mudanças das velocidades das juntas às mudanças de velocidades do atuador final. A equação que formaliza essa relação é encontrada calculando o modelo diferencial da eq. (2.17):

$$\dot{\vec{x}}(t) = J(\vec{\theta})\dot{\vec{\theta}}(t). \tag{2.24}$$

Sendo,

$$J(\vec{\theta}) = \frac{\delta f_i(\vec{\theta})}{\delta \vec{\theta}_j}$$

a Jacobiana de $f(\vec{\theta})$.

Para um sistema com $\dot{\vec{x}} \in \mathbb{R}^m$e $\dot{\vec{\theta}} \in \mathbb{R}^n$, temos:

$$J(\vec{\theta}) = \begin{bmatrix} \frac{\delta f_1(\vec{\theta})}{\delta \vec{\theta}_1} & \dots & \frac{\delta f_1(\vec{\theta})}{\delta \vec{\theta}_n} \\ \dots & \ddots & \dots \\ \frac{\delta f_m(\vec{\theta})}{\delta \vec{\theta}_1} & \dots & \frac{\delta f_m(\vec{\theta})}{\delta \vec{\theta}_n} \end{bmatrix}, \; J(\vec{\theta}) \in \mathbb{R}^{m \times n}.$$

Essa matriz possui diversas aplicações na robótica [14, 13, 15]:

- Está na base do modelo diferencial inverso;
- Facilita o cálculo de singularidades e da dimensão do espaço operacional acessível do robô.

Uma singularidade ocorre quando a velocidade das juntas no seu espaço se torna muito alto para manter a velocidade no espaço cartesiano.

Configurações singulares não podem ser reproduzidas pelo robô. Podemos encontrá-las através da Jacobiana. Quando $det(J(\vec{\theta})) = 0$, a Jacobiana associada é dita singular.

A partir da equação (2.24), podemos buscar uma solução iterativa para a equação (2.17) a partir de mudanças incrementais:

$$d\vec{x} = J(\vec{\theta})d\vec{\theta} \tag{2.25}$$

Essa relação será melhor explorada na próxima seção.

Cinemática Inversa

A relação da equação (2.25) pode ser invertida de forma semelhante à cinemática inversa (2.23), para $n = m$:

$$\dot{\vec{\theta}} = J^{-1}(\vec{\theta})\dot{\vec{x}} \tag{2.26}$$

Podemos fazer o mesmo para a equação (2.26), obtendo:

$$d\vec{\theta} = J(\vec{\theta})d\vec{x} \tag{2.27}$$

A Pseudo-Inversa

A equação (2.26) nos traz o problema de inversão da matriz Jacobiana, que nem sempre é uma matriz quadrada. Nesse caso, utilizaremos a pseudo-inversa da Jacobiana ($J^{\dagger}(\vec{\theta})$). Então, podemos reescrever (2.26) como:

$$\dot{\vec{\theta}} = J^{\dagger}(\vec{\theta})\dot{\vec{x}}. \tag{2.28}$$

A pseudo-inversa é a generalização da matriz inversa no caso em que a matriz a ser invertida não é quadrada. Não há uma unicidade desse tipo de operação. Alguns tipos de pseudo inversa são a pseudo-inversa de Moore-Penrose e a pseudo-inversa de Levemberg Marquardt.

A pseudo-inversa de Moore-Penrose é o tipo mais conhecido de matriz pseudo-inversa. A solução obtida vem da minimização a norma das velocidades das juntas. Ela é dada por:

$$J^{\dagger} = J^{T}(JJ^{T})^{-1} \tag{2.29}$$

A pseudo-inversa de amortecida é uma solução numérica para a Jacobiana. Ela evita grande parte dos problemas com singularidades que outros métodos de pseudo-inversa podem dar [16]. Sendo $k \in \mathbb{R}$ uma constante de amortecimento não nula, a pseudo-inversa amortecida é dada por:

$$J^{\dagger} = J^{T}(JJ^{T} + k^{2}I)^{-1} \tag{2.30}$$

Na Seção seguinte falaremos sobre as técnicas para inversão da matriz Jacobiana e sobre o controle implementado no robô.

2.4 Controle Cinemático

Imaginemos um manipulador que foi iniciado em uma certa pose. A partir da informação dos *encoders* dos motores, sabemos os valores de cada junta e os armazenamos em $\vec{\theta}(0)$. Usando a equação 2.17, supostamente já encontrada de acordo com a notação de Denavit-Hartenberg, podemos encontrar a posição e a translação do atuador final. Colocamos essas informações em $\vec{x}(0)$.

Agora queremos enviar o atuador final a uma nova pose descrita por $\vec{x}_{\mathrm{d}}$, a posição desejada.

Segundo [11], as posições das juntas podem ser obtidas integrando as suas velocidades no tempo:

$$\vec{\theta}(t) = \int_{0}^{t} \dot{\theta}(\varsigma)d\varsigma + \vec{\theta}(0) \tag{2.31}$$

Sendo $\vec{\theta}(t)$ o vetor com valor das juntas no tempo t e $\vec{\theta}(0)$ o vetor com os valores inicias das juntas.

Podemos então aplicar o método de integração de Euler para discretizar o sistema descrito por (2.31):

$$\vec{\theta}_{k+1} = \vec{\theta}_{k} + \dot{\vec{\theta}}_{k}\Delta t \tag{2.32}$$

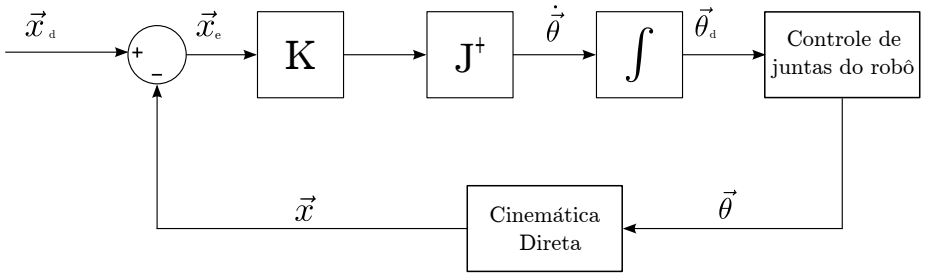


Figura 2.7: Esquema de controle com a pseudo-inversa da Jacobiana.

Tal que Δt seja o intervalo de integração; $\vec{\theta}_{k+1}$, o valor calculado para a posição desejada das juntas; $\vec{\theta}_k$ a posição atual; e $\dot{\vec{\theta}}_k$, a velocidade atual das juntas.

Caso o DOF do espaço de juntas seja diferente do DOF do espaço cartesiano, devemos usar a pseudo-inversa da Jacobiana para calcular $\dot{\vec{\theta}}_k$. Para $\vec{x}_e = \vec{x}_{k+1} - \vec{x}_k$, sendo $\vec{x}_{k+1}$ o valor desejado ($\vec{v}_d$) e $\vec{x}_k$ o valor medido ($\vec{x}$), [11] demonstrou que podemos rescrever $\dot{\vec{\theta}}_k$ da seguinte forma:

$$\dot{\vec{\theta}}_{k+1} = J^\dagger(\vec{\theta}_k)(\dot{\vec{x}}_{k+1} + K\vec{x}_e). \tag{2.33}$$

Sendo essa lei de controle assintóticamente estável [17]. A matriz de ganho K, usualmente diagonal, foi acrescentada para controlar a taxa de convergência.

Como desejamos que a velocidade final seja nula, fazemos $\dot{\vec{x}}_{k+1} = 0$. Assim, subsituindo (2.34) em (2.32) e fazendo $J^\dagger = J^\dagger(\vec{\theta}_k)$, obtemos:

$$\vec{\theta}_{k+1} = \vec{\theta}_k + J^\dagger K\vec{x}_e. \tag{2.34}$$

A Figura 2.7 apresenta o diagrama de blocos do sistema de controle implementado no robô. A cada iteração, o sistema recebe os valores das juntas do robô e a posição desejada no plano cartesiano. A partir dai, aplica-se a equação de cinemática direta para encontrar o valor equivalente da posição atual no espaço tridimensional. Aqui podemos aplicar (2.33) para encontrarmos a velocidade das juntas. Finalmente aplicamos uma integral para encontrarmos os valores desejados para as juntas.

2.5 Equilíbrio Simples

Para manter um corpo em equilíbrio estático, devemos nos assegurar que seu centro de massa está sobre o polígono de suporte do mesmo. Podemos considerar o centro de massa (CoM[6]) como o ponto no espaço onde toda a massa do corpo está concentrada. Ele não precisa estar dentro do objeto. Considerando um componente de n partículas, a posição do seu CoM é definida pela equação (2.35), onde m_i corresponde à massa da partícula i e $\vec{r}_i$ a sua posição em relação a um *frame* de referência.

$$\vec{CoM} = \frac{\sum_{i=1}^{n} m_i\vec{r}_i}{\sum_{i=1}^{n} m_i}. \tag{2.35}$$

[6]CoM: do inglês, *Center of Mass* (Centro de Massa).

O polígono de suporte é definido pelo envelope convexo. Tomando como base um humanoide em pé, o envelope é definido pelo perímetro interior de um elástico colocado em torno dos pés. No caso de o robô estar sobre um dos pés, o polígono de suporte é a própria superfície desse pé.

2.6 O Sensor de Movimento Kinect

Kinect é um sensor de movimento desenvolvido pela empresa Microsoft[7] junto à Prime Sense[8] para a consola Xbox 360. O equipamento provê um Inteface Natural ao Usuário (do inglês, *Natural User Interface*, NUI), núcleo da API[9] do Kinect para Windows, que permite interagir de forma intuitiva com um aplicativo, ou jogo, sem a necessidade de um controle (*joystick*). Ele oferece captura em três dimensões de corpo todo, reconhecimento facial e de voz.

A Figura 2.8 nos apresenta as principais partes do aparelho. O sensor de profundidade é constituido de duas partes: um laser infravermelho e um sensor CMOS. A câmera RGB (*Red*, *Green*, *Blue*) permite o reconhecimento facial e o cruzamento de informações com o sensor anterior para criação de uma imagem 3D. Um jogo de engrenagens permite seguir um usuário durante o jogo. O vetor de áudio consegue filtrar ruídos externos, detectar várias pessoas na sala, captar vozes mais próximas e assim reconhecer comandos de voz.

Figura 2.8: O Kinect, seus sensores e atuador (fonte: Microsoft)

O aparelho possui a capacidade de reconhecer e interpretar gestos específicos através da utilização da câmera RGB, da ótica infravermelha e de um microchip especial que rastrea pessoas e objetos em 3D. Essa sua última propriedade será explorada neste trabalho para capturar o movimento humano e assim criar uma trajetória de movimentação para o robô. É possível o re-

[7]Microsoft: multinacional americana especializada nos setores de computação e de informática. Suas aticidades principais incluem o desenvolvimento e venda de sistemas operacionais e de softwares.

[8]Prime Sense: antiga empresa israelense especializada em sensores 3D, foi comprada em 2013 pela Apple (multinacional americana que produz e comercializa produtos eletrônicos).

[9]API: do inglês *Application Programming Interface* (Interface de Programação do Aplicativo)

conhecimento de até 6 jogadores, mas o rastreio de movimento de apenas 2 deles. Um aplicativo consegue localizar as juntas dos usuários ativos com a detecção de até 20 juntas (Figura 2.9).

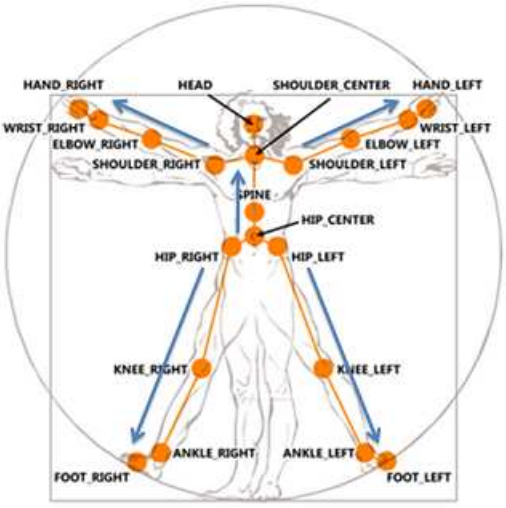


Figura 2.9: Imagem do esqueleto provido pelo Kinect e nomes específicos dados a cada junta

2.7 O Humanóide NAO

O NAO é um robô humanóide programável desenvolvido pela empresa francesa Aldebaran Robotics. O pequeno humanóide de 58cm de altura nasceu em 2006 a partir da projeto de criação de robôs humanóides que pudessem fazer companhia ao homem.

Muito conhecido no meio acadêmico, sua plataforma robusta permite o desenvolvimento de aplicações de *software* com a abstração do desenvolvimento de *hardware*. Essa é a ideia principal da *Standard Plataform League*, uma categoria da *Robocup* que permite apenas a utilização de humanóides NAO para uma competição de futebol (Figura 2.10). Nessa liga de futebol, as equipes devem criar o melhor *software* para que o robô jogue futebol de forma autônoma e em equipe. Isso deve ser feito sem que haja qualque modificação nos sensores, atuadores, ou qualquer outra parte elétrica, ou mecânica do NAO.

Trataremos melhor suas especificações na Seção 3.3.

2.8 Conclusão

Vimos nesse capítulo os aspectos teóricos usados para modelação e controle do robô. No próximo capítulo, veremos como utilizamos esses recursos para movimentarmos um humanóide, de acordo com as informações adquiridas do Kinect.

Figura 2.10: Robôs humanóides da Standard Platform League, da Robocup (fonte: Paristech).

Capítulo 3

Desenvolvimento

Os aspectos fundamentais para a concepção e controle de um robô incluem a modelagem, o planejamento e o controle [11]. No capítulo anterior apresentamos as ferramentas matemáticas que nos permitem modelar uma cadeia cinemática e controlar seu posicionamento. A etapa de planejamento consta na captura e tratamento de dados do kinect. Neste capítulo veremos as ferramentas utilizadas para concretizar o controle do projeto e como ele foi desenvolvido.

3.1 Introdução

Para deixar mais claro o fluxo do projeto, a Figura 3.1 nos apresenta o diagrama de blocos para o controle de posição dos atuadores finais do NAO.

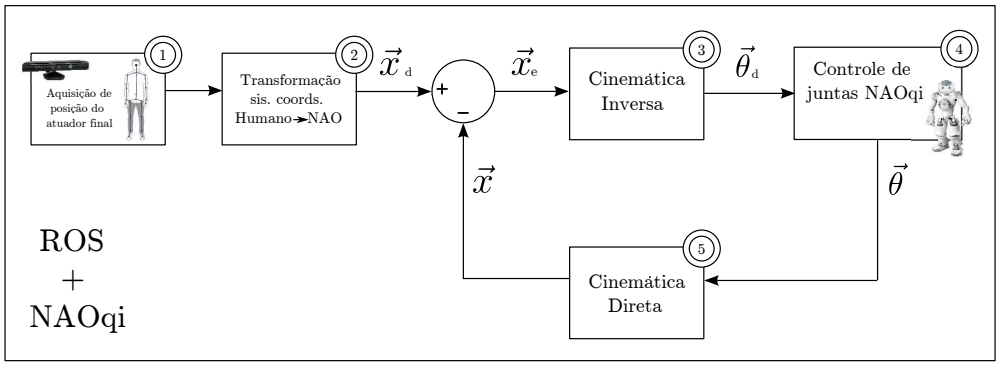


Figura 3.1: Esquema de controle de movimentação do NAO.

Na figura, representamos cada etapa do processo com um bloco identificado por um número. Temos então os seguintes passos:

1. Aquisição da posição do atuador final: temos 4 atuadores finais (2 mãos e 2 pés) que são rastreados pelo Kinect. O sensor é capaz de dar a posição de cada uma dessas partes no plano tri-dimensional;

2. Transformação da posição do sistema de coordenadas do humano para o do NAO: como o

robô e o humano não possuem as mesmas dimensões, devemos transformar o dado adquirido de cada efetuador-final do humano para o plano cartesiano do robô;

3. Cinemática Inversa: tendo os dados da posição desejada ($\vec{x}_d$) e da posição atual ($\vec{x}$) do atuador em questão, podemos descobrir os valores das juntas, $\vec{\theta}_d$, que correspondem à posição desejada.

4. Controle de juntas NAOqi: enviamos os dados dos ângulos das juntas para o *software* controlador do robô, provocando o movimento do manipulador em questão;

5. Cinemática Direta: a partir de $\vec{\theta}$, podemos encontrar a posição atual, $\vec{x}$, do robô. Esse dado é usado para realimentação negativa do controlador.

Utilizaremos duas abordagens para a cinemática inversa e controle cinemático. Na primeira, utilizaremos comandos pré-existentes do *framework* próprio do NAO para controlar a posição do efetuador-final. Na segunda, utilizaremos um *framework* especial para implementarmos um controle baseado em quatérnios duais.

Voltando à Figura 3.1, no canto inferior esquerdo, indicamos os ambientes onde o trabalho foi desenvolvido: ROS e NAOqi. Essas ferramentas de *software* e todas as etapas do processo descritas brevemente acima serão apresentadas e discutidas neste capítulo.

Inicialmente apresentaremos o ambiente de trabalho para desenvolvimento do código e os *frameworks* utilizados. A Seção 3.3 apresenta como é feita a modelagem cinemática do NAO para representação de transformações. Na Seção 3.4, explicaremos como foram adquiridos e tratados os dados de rastreio do Kinect e entenderemos como foram implementadas as duas abordagens propostas para controle de movimentação das cadeias cinemáticas.

3.2 Ferramentas de Software

Para a nossa concepção de implementação para o problema dado, utilizamos o ambiente de programação do ROS, sistema que facilita a criação de aplicações para a robótica. Para o controle da movimentação do robô e leitura de sensores, utilizamos o NAOqi. Antes de testar os comandos diretamente no robô, utilizamos o simulador Webots for NAO. Apresentaremos brevemente essas ferramentas a seguir.

3.2.1 NAOqi

NAOqi[1] é o *software* principal que roda no robô. Seu framework permite a programação e o controle do NAO. Ele pode ser explorado em diferentes plataformas (*cross plataform*): Windows, Linux e Mac. Além disso, ele oferece uma API que permite a utilização de C++, ou Python, como linguagens de programação do robô (*cross-language*).

[1]NAOqi: mais informações podem ser encontradas em https://community.aldebaran.com/doc/1-14/dev/naoqi/index.html

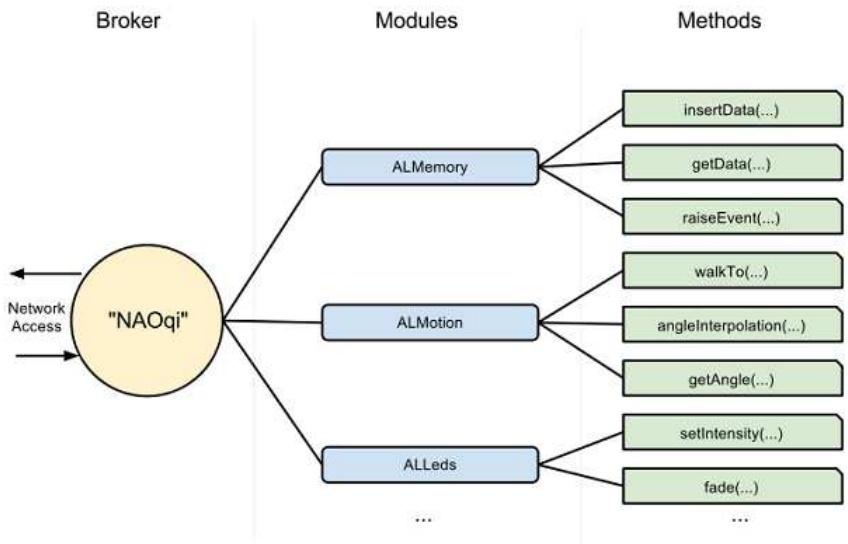


Figura 3.2: Árvore de métodos relacionados a módulos, por sua vez relacionados ao broker, que provê acesso à rede e serviços para encontrar módulos e métodos.

No robô, há um NAOqi executável que age como um mediador, para a comunicação entre aplicações. Ele é chamado de *broker*. Ao ser iniciado, ele carrega as bibliotecas que serão usadas. Essas contêm os módulos, que utilizam o *broker* para conseguir trocar mensagens entre si. Cada módulo, por sua vez, possui diversos métodos (getAngle(), walkTo(), etc.). A Figura 3.2 nos apresenta a árvore formada pela relação entre *broker*, módulos e métodos.

3.2.2 ROS

ROS[2] é uma abreviação de *Robotic Operational System*. Ele foi desenvolvido pela sociedade americana Willow Garage[3] inicialmente apenas para integrar funcionalidades do PR2 (robô desenvolvido pelo mesmo grupo). Hoje ele se extende a outras plataformas robóticas como o NAO, o Q.bo[4], entre outros, além de oferecer grandes facilidades na integração de ferramentas computacionais para robôs. Utilizamos neste trabalho a última versão liberada do ROS, a distribuição ROS Hydro Medusa.

O ROS oferece bibliotecas e ferramentas para integralizar funcionalidades e facilitar o desenvolvimento de aplicações robóticas. Sua licença é open source (licença BSD[5]).

Algumas das ferramentas mais importantes do ROS, utilizadas neste trabalho são brevemente listadas a seguir.

O *framework* pode ser dividido em três níveis:

- nível de arquivos do sistema (basicamente os recursos que estão no disco, como pacotes,

[2]ROS: mais informações podem ser encontradas em http://www.ros.org/
[3]Willow Garage: laboratório de pesquisa em robótica criado em 2006.
[4]Q.bo: robô móvel desenvolvido pela sociedade espanhola The Corpora.
[5]BSD: do inglês, *Berkeley Software Distribution license*. Ela permite a utilização parcial, ou total de um *software*.

meta-pacotes, mensagens, serviços.);

- nível de computação gráfica (alguns de seus conceitos são os nós, o Master, as mensagens, os serviços);
- nível de comunidade (recursos que permitem à comunidade de usuários do ROS trocar dados e conhecimentos sobre o mesmo).

No nível de computação gráfica, temos alguns conceitos importantes a destacar. O Master gerencia o registro de nomes, além de registrar as informações de nós e serviços. É através dele que os nós (processos) se tornam cientes da existência de outros nós e podem de comunicar. A comunicação ocorre através de tópicos (sistema com publicador e assinante) e de serviços (sistema de requisição e resposta). Os nós se comunicam pelo envio e recepção de mensagens

3.2.3 Webots for NAO

O Webots[6] é um simulador profissional de robótica desenvolvido pela empresa suiça Cyberbotics Ltd. Ele permite a modelagem, programação e simulação de robôs. O Webots for NAO é uma versão desse simulador desenvolvida para controlar apenas o NAO. Nesta plataforma, outros robôs não estão disponíveis. Podemos observar sua interface gráfica na Figura 3.3.

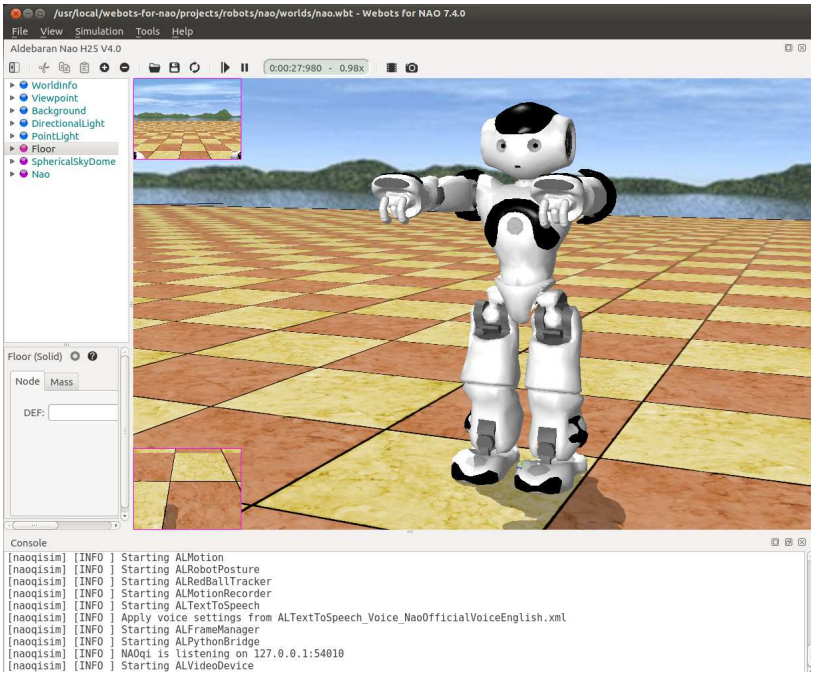


Figura 3.3: Interface gráfica do simulador Webots for NAO

Com o uso do simulador, é possível tratar dados dos seus sensores e enviar informações aos seus atuadores. O robô pode ser colocado em diferentes ambientes, pode estar sujeito a forças externas. O simulador permite que objetos possam ser colocados no ambiente, para interação com o robô.

[6]Webots: mais informações podem ser encontradas em http://www.cyberbotics.com/overview.

O controle do robô é feito a partir de um programa externo que utiliza instruções do NAOqi. Dessa forma, à princípio, a programação testada no NAO simulado é a mesma no real.

Novos comandos eram sempre testados inicialmente na simulação, antes de serem implementados no modelo real.

3.3 Especificações do Humanóide NAO

Devemos compreender algumas características de hardware do robô antes de seguir em frente. As informações a seguir foram todas retiradas do site da empresa fabricante Aldebaran.

3.3.1 Especificações do Robô

O NAO V4 H25 é o modelo do humanóide usado para nossos experimentos. Ele possui um computador embarcado com um processador x86 Intel ATOM Z530, com frequência de *clock* de 1,6Ghz, 1GB RAM, 512kB de memória cache do tipo L2, 2GB de memória flash e um MICRO SDHC de 8GB.

Uma gama de sensores permite fácil interação do NAO com o ambiente externo. A Figura 3.4 nos apresenta a disposição de todos os sensores do robô. Ele possui duas caixas de som e um microfone, podendo receber e emitir informações de áudio. Suas duas câmeras estão montadas na cabeça sobre o eixo vertical do robô. Suas imagens não se superposicionam, não sendo possível o processamento de visão estereoscópica. Há dois sonares na parte frontal do tronco (dois emissores e dois receptores). Ele também possui sensores inerciais: um girômetro e um acelerômetro. Quatro sensores sensíveis a força (FSR) estão localizados em baixo de cada pé. Temos *encoders* magnéticos de rotação em cada motor e *bumpers* nas mãos e nos pés.

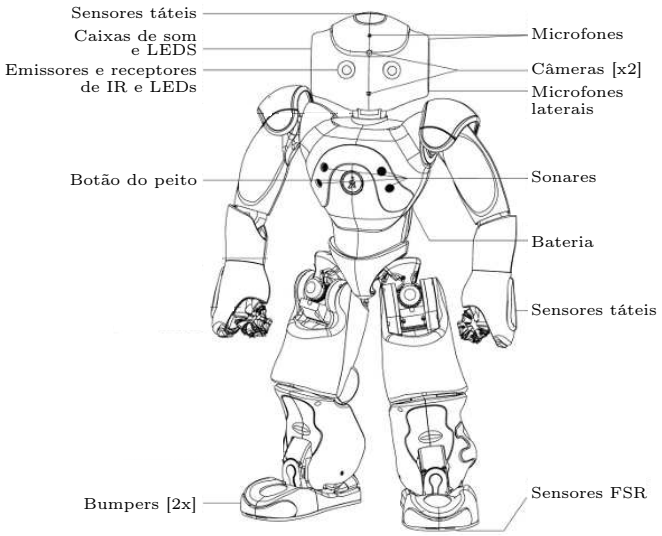


Figura 3.4: Distribuição dos sensores no corpo do humanóide NAO (fonte: Aldebaran)

O NAO pode ser dividido em 5 cadeias cinemáticas. A Tabela 3.1 apresenta as juntas pertencentes a cada cadeia cinemática e Figura 3.5 apresenta a distribuição delas no robô. Uma particularidade sobre as juntas ocorre na região pélvica, onde há um grau de liberdade "arfGuiQuadril" que provoca uma movimentação espelhada das pernas. Assim, o comando para o movimento dessa junta irá movimentar as duas pernas ao mesmo tempo.

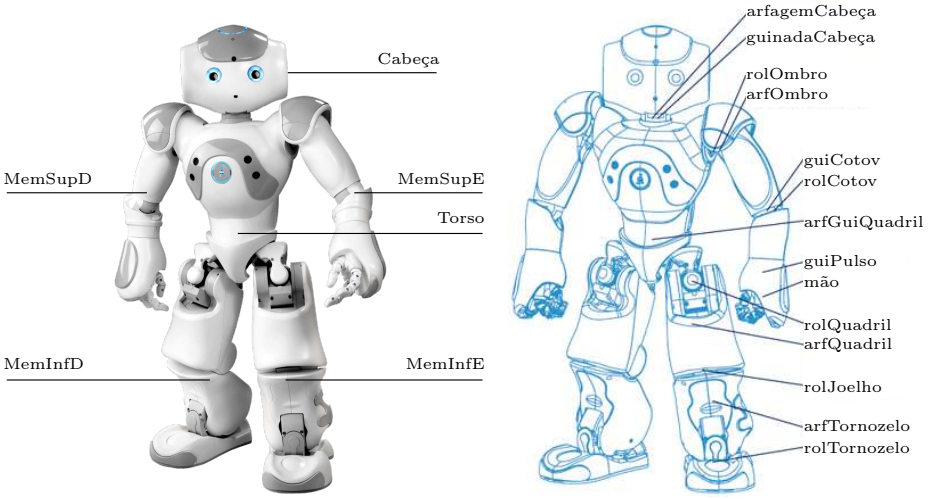


Figura 3.5: Cadeias cinemáticas do humanóide NAO e suas juntas (fonte: Aldebaran).

Cadeia	Juntas
Cabeça	arfagemCabeça - guinadaCabeça
MemSupD	rolOmbroD - arfOmbroD - guiCotoveloD - rolCotovD - guiPulsoD - mãoD
MemSupE	rolOmbroE - arfOmbroE - guiCotoveloE - rolCotovE - guiPulsoE - mãoE
MemInfD	arfQuadrilD - rolQuadrilD - arfJoelhoD - rolJoelhoD- arfTornozeloD - rolTornozeloD
MemInfD	arfQuadrilE - rolQuadrilE - arfJoelhoE - rolJoelhoE - arfTornozeloE - rolTornozeloE

Tabela 3.1: Cadeias cinemáticas do NAO e as juntas que as formam.

Para os próximos passos, é importante ainda que tenhamos alguns dados sobre as dimensões do robô. A Figura 3.6 apresenta algumas dessas dimensões..

3.3.2 Parâmetros de Denavit-Hartenberg do NAO

Para encontrar a equação de cinemática direta do robô usamos a convenção de Denavit-Hartenberg modificada (Seção 2.3). Todos os parâmetros foram recuperados do site da Aldebaran.

A Figura 3.7 apresenta as transformações feitas em cada junta, em relação à anterior, para obtermos os parâmetros de Denavit-Hartenberg. A posição zero do robô, onde os valores das juntas estão em zero, ocorre quando ele está em pé com seus braços para frente, como na Figura

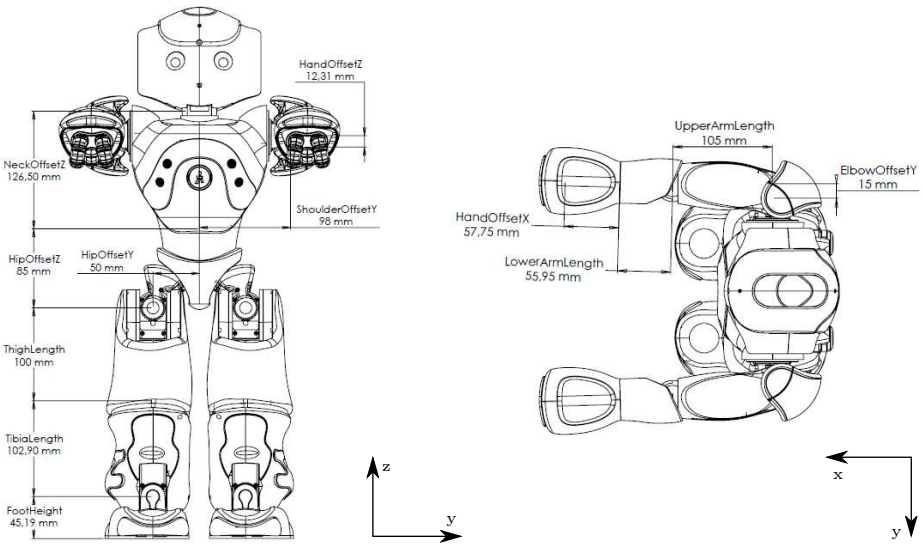


Figura 3.6: Dimensões do NAO. À esquerda temos a sua visão frontal e à direita, sua visão superior. (Fonte: Aldebaran)

3.6. Consideramos aqui as cadeias cinemáticas dos membros superiores e dos membros inferiores. Para este trabalho, ignoramos a cadeia cinemática da cabeça.

As Tabelas 3.2 e 3.3 apresentam os parâmetros referentes a cada junta. A Tabela 3.4 apresenta as transformações que devem ser aplicadas para mover a base do sistema de coordenadas do torso para a base da cadeia cinemática desejada. A Tabela 3.5 nos mostra a transformação final para translacionar o sistema de coordenadas ao final físico da cadeia e para rotacioná-lo de forma que ele corresponda à rotação inicial do torso.

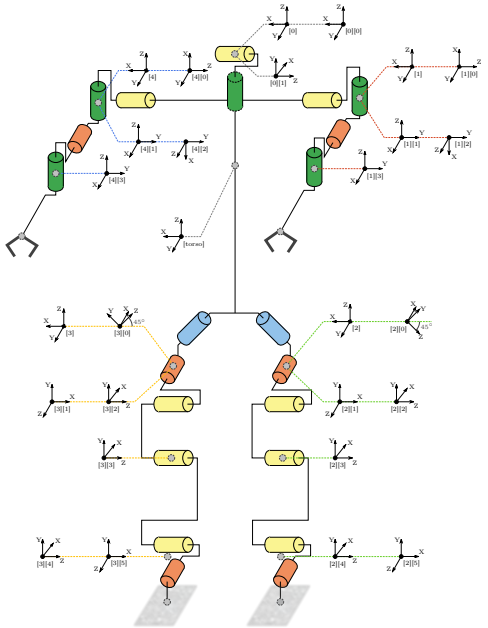


Figura 3.7: Esquema com parâmetros de Denavit-Hartenberg do corpo do humanóide NAO (criado por: Stefan Glaser).

Junta	α (rad)	a $(metros)$	θ (rad)	d $(metros)$
Arfagem do ombro (T_1^0)	$-\frac{\pi}{2}$	0.0	0.0	0.0
Rolagem do ombro (T_2^1)	$\frac{\pi}{2}$	0.0	$\frac{\pi}{2}$	0.0
Guinada do cotovelo (T_3^2)	$\frac{\pi}{2}$	0.0	0.0	tBraço
Rolagem do cotovelo (T_4^3)	$-\frac{\pi}{2}$	0.0	0.0	0.0
Guinada do pulso (T_5^4)	$\frac{\pi}{2}$	0.0	0.0	tAntebraço

Tabela 3.2: Parâmetros de Denavit-Hartembert para membros superiores (convenção modificada)

Junta	α (rad)	a $(metros)$	θ (rad)	d $(metros)$
Guinada/Arfagem do quadril (T_1^0)	$-\frac{\pi}{4}$	0.0	$-\frac{\pi}{2}$	0.0
Rolamento do quadril (T_2^1)	$-\frac{\pi}{2}$	0.0	$-\frac{\pi}{4}$	0.0
Arfagem do quadril (T_3^2)	$\frac{\pi}{2}$	0.0	0.0	0.0
Arfagem do joelho (T_4^3)	0.0	- tCoxa	0.0	0.0
Rolamento do tornozelo (T_5^4)	0.0	- tTíbia	0.0	0.0
Arfagem do tornozelo (T_6^5)	$-\frac{\pi}{2}$	0.0	0.0	0.0

Tabela 3.3: Parâmetros de Denavit-Hartembert para membros inferiores (convenção modificada)

Cadeia	Transformada para a base ($T_{\text{início}}$)
Membro superior esquerdo	$T_{trans,y}$(offsetOmbroY)$T_{trans,z}$(offsetOmbroZ)
Membro superior direito	$T_{trans,y}$(-offsetOmbroY)$T_{trans,z}$(offsetOmbroZ)
Membro inferior esquerdo	$T_{trans,y}$(offsetQuadrilY)$T_{trans,z}$($-$offsetQuadrilZ)
Membro inferior direito	$T_{trans,y}$(-offsetQuadrilY)$T_{trans,z}$($-$offsetQuadrilZ)

Tabela 3.4: Transformadas da base do sistema de coordenadas de referência (torso), para a base das cadeias cinemáticas

Cadeia	Transformada para o efetuador-final (T_{final})
Membro superior esquerdo	$T_{rot,x}(-\pi/2)T_{rot,y}(-\pi/2)T_{trans,x}$(offsetMãoX)$T_{trans,z}$($-$offsetMãoZ)
Membro superior direito	$T_{rot,x}(-\pi/2)T_{rot,y}(-\pi/2)T_{trans,x}$(offsetMãoX)$T_{trans,z}$($-$offsetMãoZ)
Membro inferior esquerdo	$T_{rot,z}(\pi)T_{rot,y}(-\pi/2)T_{trans,z}$($-$alturaPé)
Membro inferior direito	$T_{rot,z}(\pi)T_{rot,y}(-\pi/2)T_{trans,z}$($-$alturaPé)

Tabela 3.5: Transformadas do último ponto da cadeia cinemática para o efetuador-final

A partir desses dados, podemos encontrar as equações cinemáticas de cada cadeia. Para membros superiores e inferiores, as equações das cadeias à direita e à esquerda se diferenciam apenas em relação às transformações iniciais ($T_{\text{início}}$) e finais (T_{final}). Podemos então representar essas cadeias genericamente pelas Equações 3.1 e 3.2.

$$T_{\text{membro\_superior}} = T_{\text{início}} T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_{\text{final}} \tag{3.1}$$

e,

$$T_{\text{membro\_inferior}} = T_{\text{início}} T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_{\text{final}} \tag{3.2}$$

Para 3.1 e 3.2, encontramos as transformadas $T_{\text{início}}$ e T_{final} nas Tabelas 3.2 e 3.3. As transformações de T_1^0 a T_5^4 da equação 3.1 podem ser encontradas substituindo os valores da Tabela 3.2 na equação 3.3, que foi apresentada no capítulo anterior. As trasformações de de T_1^0 a T_6^5da equação 3.2, referentes aos membros inferiores, podem ser encontradas utilizando-se os valores da Tabela 3.3 na equação 3.3.

$$T_i^{i-1} = T_{DH} = T_{rot,z}(\theta) T_{trans,z}(d) T_{trans,x}(a) A_{rot,x}(\alpha) \tag{3.3}$$

O módulo ALMotion é o responsável por manter o sistema atualizado da posição de cada junta do robô.

3.4 Controle de Trajetória

3.4.1 Características gerais

As características descritas nesta subseção dizem respeito à logica para implementação do controle de trajetória, desde a definição da rota dos membros até o controle de movimentação do robô.

A Figura 3.8 nos dá uma visão geral da solução proposta em ROS. Basicamente temos nós que trocam mensagens entre si através de tópicos, ou serviços. O diagrama noção do fluxo que seguem as mensagens. Explicaremos melhor a função de cada item da Figura 3.8 a seguir.

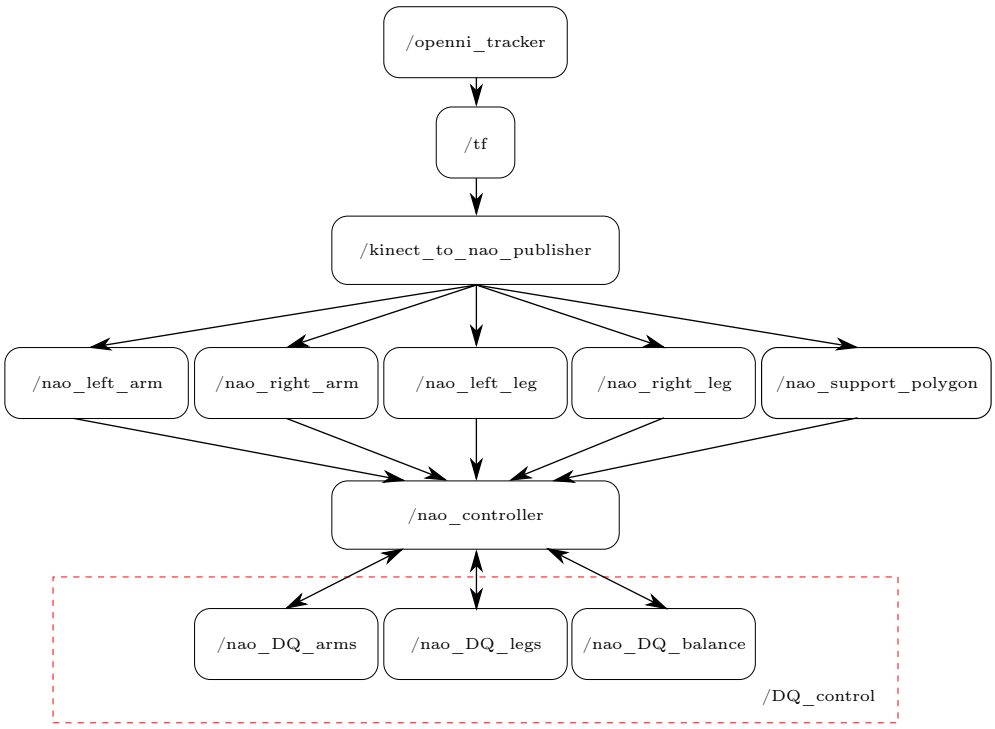


Figura 3.8: Diagrama de fluxo de dados da solução proposta para controle por teleoperação

3.4.2 Definição de Trajetória

Como foi visto na Seção 2.6, para um usuário ativo no Kinect, podemos obter a posição de até 20 de suas juntas no plano tri-dimensional. Para o nosso controle de posição estamos interessados em obter as posições dos atuadores-finais: mãos e pés.

O ROS possui diversos *drivers* para acessar os dados do Kinect[7], e.g. freenect_stack, openni_kinect, kinect e kinect_aux. O openni_kinect, em especial, recebe suporte oficial da PrimeSense. Ele engloba os *drivers* OpenNI, além de algumas bibliotecas de alto nível. Uma dessas bibliotecas oferece acesso aos dados do esqueleto do usuário (biblioteca *skeleton*). Podemos acessar suas funcionalidades através do pacote em ROS openni_tracker.

O openni_tracker publica as múltiplas coordenadas (uma para cada junta) de um usuário a cada quadro no tempo. Para isso, ela utiliza o pacote tf (*transform*) em ROS. Esse pacote possui uma série de particularidades que permitem rastrear um, ou vários quadros de coordenadas no tempo, além de dar a posição relativa (posição e orientação) do plano de uma junta em relação a um plano de referência, entre outras funcionalidades.

Para cada frame, uma série de transformadas tf serão publicadas, como podemos observar na Figura 3.9. Todas representam uma transformada em relação a um quadro fixo, chamado /openni_depth_frame. Diz-se que esse é o quadro pai e os quadros filhos são: /head; /neck; /torso; /left_shoulder; /left_elbow; /left_hand; /right_shoulder; /right_elbow; /right_hand; /left_hip; /left_knee; /left_foot; /right_hip; /right _knee; e /right_foot.

[7]Kinect no ROS: mais informações podem ser encontradas em http://wiki.ros.org/kinect

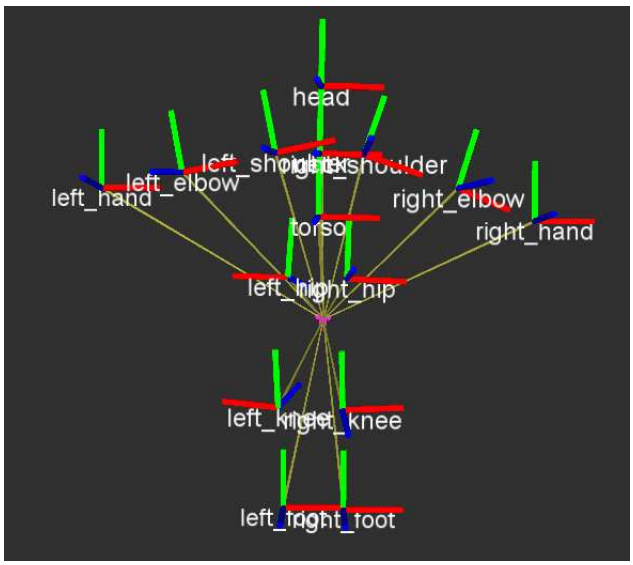


Figura 3.9: Esqueleto completo com transformadas do quadros de cada junta e seus nomes.

Voltando ao fluxo do projeto, observando o diagrama da Figura 3.8, temos o openni_tracker, que publica informações através de tf, e, em seguida, os dados são recebidos por um nó chamado kinect_to_nao_publisher. Esse processo é ao mesmo tempo um *listener* e um *broadcaster*. Basicamente ele "escuta" dados, os manipula e os difunde, quando prontos.

Em sua função de *listener*, o kinect_to_nao_publisher recebe os dados de rotação e translação de todas as juntas e os armazena. O próximo passo é passar cada pose do sistema de coordenadas do humano para o sistema de coordenadas do robô. Faremos isso através de uma mudança de escala e de translações sucessivas.

Tomemos como exemplo o tratamento de informações para o braço esquerdo. Sua base corresponde ao ombro esquerdo, com posição $\vec{h}_0$, e seu efetuador-final corresponde à mão esquerda, com posição $\vec{h}_n$. Nesse caso em que só temos 3 juntas (ombro, cotovelo e pulso/mão), $n = 2$. Para encontrarmos a posição $\vec{r}_n$, correspondente a $\vec{h}_n$ no robô, aplicamos a equação 3.4 iterativamente da base ao ponto final do manipulador e em seguida, somamos as posições $\vec{r}$ encontradas (equação 3.5) para termos $\vec{r}_n$ com referência à base do manipulador do NAO.

$$\vec{a}_{i+1} = (\vec{h}_{i+1} - \vec{h}_i) \times \frac{\|\vec{r}_{i+1} - \vec{r}_i\|}{\left\|\vec{h}_{i+1} - \vec{h}_i\right\|} \tag{3.4}$$

$$\vec{r}_n = \sum_{j=1}^{n} \vec{a}_j \tag{3.5}$$

No exemplo do braço, teríamos:

$$\vec{a}_{\text{cotovelo\_esq}} = (\vec{h}_{\text{cotovelo\_esq}} - \vec{h}_{\text{ombro\_esq}}) \times \frac{\text{tBraço}}{\left\|\vec{h}_{\text{cotovelo\_esq}} - \vec{h}_{\text{ombro\_esq}}\right\|},$$

e

$$\vec{a}_{\text{mão\_esq}} = (\vec{h}_{\text{mão\_esq}} - \vec{h}_{\text{cotovelo\_esq}}) \times \frac{\text{tAntebraço}}{\left\|\vec{h}_{\text{mão\_esq}} - \vec{h}_{\text{cotovelo\_esq}}\right\|}.$$

Finalmente,

$$\vec{r}_{\text{mão}} = \vec{a}_{\text{cotovelo\_esq}} + \vec{a}_{\text{mão\_esq}}.$$

Agora que temos a posição dos atuadores-finais desejados, devemos transladar a base do manipulador para a sua posição real no sistema de coordenadas do robô, que fica no seu torso. Os valores dessas translações estão na Figura 3.6.

Após esse tratamento de dados, podemos enviar as novas posições para o robô. O fazemos através de 4 tópicos: /nao_left_arm, /nao_right_arm, /nao_left_leg e /nao_right_leg. O quinto tópico observado na Figura 3.8 envia informação sobre em qual suporte a pessoa se encontra: pé direito, pé esquerdo, ou dois pés no chão. Falaremos sobre ele na subseção 3.4.6.

Agora que temos as posições desejadas, vejamos como são recebidos e tratados os dados pelo nó nao_controller.

3.4.3 Nao_controller

Esse nó é o responsável pelo envio de comandos para o NAO. Ele inicializa os módulos que serão usados no robô e em seguida, se subscreve aos cinco tópicos criados no nó anterior. Essa segunda operação é feita com a criação de cinco *subscribers*. Cada vez que o nó kinect_to_nao_publisher envia mensagens pelos seus tópicos, os *subscribers* do nao_controller recebem e passam essas informações a funções *callback*. Limitamos a fila do *buffer* do *subscriber* a uma mensagem por vez. Dessa forma, caso alguma nova informação chegue enquanto ele está processando, ela será dispensada.

Para cada membro que estamos rastreando, quando seu *subscriber* correspondente recebe a mensagem com a sua nova posição, uma função *callback* para o controle de posição recebe essa mensagem e seta novos ângulos aos manipuladores do NAO.

O cálculo das novas posições das juntas, a partir das posições dos efetuadores-finais, foi realizado utilizando duas abordagens. Uma delas, com uma função já existente no conjunto de instruções do NAOqi e a segunda, com quatérnios duais.

3.4.4 Abordagem de controle com instruções pré-existentes do NAOqi

Toda os processos descritos nessa subseção são realizados dentro do nao_controller. Desconsideramos aqui os blocos subsequentes, que estão dentro do retângulo vermelho tracejado.

O framework padrão da Aldebaran utiliza a biblioteca ALMath [18] para ter acesso a ferramentas de cinemática e dinâmica para o controle do NAO. ALMath reune instruções de matemática otimizada para robótica.

Para o controle de posição e orientação de uma parte do robô no espaço tridimensional, essa biblioteca utiliza a representação matricial, como foi visto na Subseção 2.2.1. Sua pose é descrita por um vetor coluna ($\vec{p}$) de seis componentes, sendo os três primeiros correspondentes à posição

do ponto em relação à referência e os três últimos, à sua rotação.

$$\vec{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ p_\alpha \\ p_\beta \\ p_\gamma \end{bmatrix} \in \mathbb{R}^6$$

Para aplicar transformações em algum ponto, são usadas as matrizes homogênas . A convenção para aplicar uma rotação em um corpo rígido definido por $\vec{p}$ deve ser $R_z(\gamma)R_y(\beta)R_x(\alpha)$.

Para controlarmos as juntas do robô dos membros superiores e inferiores, precisamos das suas equações de cinemática direta (eq. 2.17). Para encontrá-las, substituimos os valores encontrados na Sub-seção 3.3.2, na equação 2.19, para cada junta de uma cadeia cinemática e em seguida, aplicamos transformações consecutivas para ter a transformação completa da base da cadeia até o seu efetuador-final. A partir desses dados, podemos encontrar a equação de cinemática inversa correspondente (eq. 2.26).Para o cálculo da inversa da Jacobiana, a ALMath utiliza a pseudo-inversa de Moore-Penrose (eq. 2.29).

Dentre as funções de cinemática inversa do NAOqi, escolhemos uma função não-bloqueante para calcular e setar os novos ângulos das juntas dos manipuladores. Isso significa que, caso cheguem novas instruções (nova posição, ou uma outra função de controle), essa função pode ser interrompida. Veremos os resultados dessa implementação no próximo capítulo.

3.4.5 Abordagem de controle com quatérnios duais

Para a criação e manipulação de quatérnios duais, utilizaremos o *framework* DQRobotics[8]. No diagrama da Figura 3.8, temos um retângulo vermelho tracejado que engloba ferramentas para controle com quatérnios duais. Observamos a existência de três nós: nao_DQ_arms, nao_DQ_legs e nao_DQ_balance. Esse último nó será explicado posteriormente. Trataremos os outros dois nós genericamente por nao_DQ_X, sendo $X \in \{\text{arms, legs}\}$.

De acordo com o que vimos no esquema da Figura 2.7, para controlar a movimentação de um dos membros, precisamos de sua posição atual ($\vec{x}$) e da posição desejada ($\vec{x}_d$).

Utilizamos então o sistema de requisição e resposta do ROS para trocar informações entre o nao_controller e os nós nao_DQ_arms, ou nao_DQ_legs. O nao_controller envia os valores atuais das juntas ($\vec{\theta}$), com os que encontramos $\vec{x}$ por cinemática direta, conforme 2.24, e a posição desejada ($\vec{x}_d$). Em seguida, ele aguarda as posições $\vec{\theta}_d$ dos ângulos das juntas, como resposta de nao_DQ_arms, ou nao_DQ_legs.

A posição $\vec{x}_d$ chega como um ponto no sistema de coordenadas do NAO, com base no torso. Então, inicialmente o nao_DQ_X translada a base do sistema de referência do ponto desejado

[8]DQRobotics: biblioteca com instruções para controle e manipulação de quatérnios duais, para aplicação na robótica.

para a base da cadeia cinemática em questão. Em seguida, é criado um quatérnio dual, $\underline{\mathbf{x}}_d$, para representar esse ponto. Para isso, usamos 2.16, para translação pura de um quatérnio dual.

O próximo passo é encontrar a equação de cinemática direta para termos a posição atual $\underline{\mathbf{x}}$. De forma análoga ao que foi realizado com matrizes, encontramos a transformação $\underline{\mathbf{q}}_{DH}$ para cada junta com as informações da Subseção 3.3.2, e em seguida encontramos $\underline{\mathbf{q}}_i^0$, que consiste na composição das transformações de cada junta em relação a anterior da base ao efetuador-final.

Podemos agora encontrar a posição atual do efetuador-final através da equação de cinemática direta:

$$\underline{\mathbf{q}}_i^0(\vec{\theta}) = \underline{\mathbf{x}}.$$

Estamos interessados em ajustar apenas a translação do membro, ignorando a rotação. Assim, desconsideraremos a parte de rotação das mãos, ou dos pés, para os cálculos de $\vec{\theta}$.

O próximo passo agora é calcular o erro de translação entre o ponto desejado e o ponto atual.

Para extrair o quatérnio $\mathbf{t}$ referente à translação de um quatérnio dual $\underline{\mathbf{x}}$, verificamos a eq. 2.15. Dela, extraimos que:

$$\mathbf{t} = 2 \times \mathrm{D}(\underline{\mathbf{x}}) \times \mathrm{P}(\underline{\mathbf{x}})^{-1},$$

sendo $\mathrm{D}(\underline{\mathbf{x}})$, a parte dual de $\underline{\mathbf{x}}$ e $\mathrm{P}(\underline{\mathbf{x}})$, a parte primária de $\underline{\mathbf{x}}$. Podemos dizer ainda que $\mathrm{P}(\underline{\mathbf{x}})^T = \mathrm{P}(\underline{\mathbf{x}})^{-1}$, já que $\|\underline{\mathbf{x}}\| = 1$.

Calculamos então o erro de translação:

$$\vec{t_e} = vect(\mathbf{t}_d - \mathbf{t}).$$

Caso a norma desse valor seja maior que um certo limiar escolhido anteriormente, devemos calcular a nova posição do efetuador-final. Para isso, calculamos a jacobiana analítica apenas da translação [17], [19]. Para o cálculo da pseudo-inversa, optamos por utilizar a pseudo-inversa amortecida (eq. 2.30). Finalmente, obtemos $\triangle\vec{\theta}_d$ a partir do cálculo de $J^{\dagger}_{\text{trans}} K_{\text{trans}} \vec{t_e}$. Logo, encontramos $\vec{\theta}_d = \vec{\theta} + \triangle\vec{\theta}_d$.

Essa informação é então enviada de volta ao nó_NAO que seta o comando para que as juntas se movimentem de acordo com $\vec{\theta}_d$. O pseudo-código desse algorítimo está em Algorithm 1.

3.4.6 Equilíbrio

Assim como para o controle de trajetória, utilizamos duas abordagens para controlar o equilíbrio do humanóide quando ele está sobre apenas um de seus pés. Basicamente o NAO pode estar em 3 estados: duplo-suporte (em cima de ambos os pés); suporte direito (pé direito); e suporte esquerdo (pé esquerdo).

A detecção de mudança do duplo-suporte para o suporte de algum dos pés é feita no kinect_to_nao_publisher, quando é calculada uma diferença significante de altura entre os dois pés. A informação sobre qual é o atual polígono de suporte que o humano está é enviada pelo tópico nao_suport_poligon.

Algorithm 1 Cinemática Inversa Usando Quatérnios Duais

procedure CINEMÁTICA_INVERSA_DQ($cadeia$, $\vec{\theta}$, $\vec{x_d}$, k_{trans}, K_{trans})
  obter cinemática da cadeia com parâmetros de Denavit-Hartenberg
  $\underline{\mathbf{x}} \leftarrow aplicar\_cinematica\_direta(\vec{\theta})$
  transladar base de $\vec{x_d}$ para a base da cadeia com $T_{\text{início}}$
  converter $\vec{x_d}$ em um quatérnio dual, chamá-lo $\underline{\mathbf{x_d}}$
  $\vec{t_e} \leftarrow translacao(\underline{\mathbf{x}}) - translacao(\underline{\mathbf{x}}_d)$ ▷ Erro de translação. Retorna um vetor de 4 dimensões.
  if $norma(\vec{t_e}) < limiar$ **then**
    $J \leftarrow obter\_jacobiana\_analitica(\vec{\theta})$
    $J_{\text{trans}} \leftarrow obter\_jacobiana\_de\_translacao(J, \underline{\mathbf{x}})$
    $J^{\dagger}_{\text{trans}} \leftarrow J^{T}_{\text{trans}}(J_{\text{trans}}J^{T}_{\text{trans}} + k^{2}_{\text{trans}}I)^{-1}$ ▷ Pseudo-inversa amortecida.
    $\delta\vec{\theta} \leftarrow J^{\dagger}_{trans} * K_{trans} * \vec{t_e}$
    $\vec{\theta}_d \leftarrow \vec{\theta} + \delta\vec{\theta}$
  end if
  return $\vec{\theta}_d$
end procedure

No nao_controller, uma função de callback é chamada a cada mudança de suporte. Quando o robô está sobre os dois pés e ele é informado que deve mudar para o suporte direito, ou esquerdo, ele pode fazer isso diretamente, sem estados intermediários. Ele então desloca seu centro de massa para que sua projeção fique dentro do polígono de suporte definido pelo pé direito, ou esquerdo. Em seguida ele começa a seguir a trajetória do pé levantado.

Caso o NAO esteja sobre um dos pés e ele receba a instrução para passar seu suporte para o outro pé, ele deve obrigatoriamente passar pela posição de duplo-suporte antes. Garantindo maior estabilidade. Podemos ver essas fases na Figura 3.10.

Figura 3.10: Diagrama de estados para função de equilíbrio.

A primeira abordagem (apenas com instruções do NAOqi) utiliza a função de equilíbrio do *software* padrão. A segunda abordagem veio da necessidade de termos mais controle sobre essa função de equilíbrio.

Ela se baseia em manter o equilíbrio estático do robô a cada pequeno mudança de posicionamento do corpo do NAO. Fazemos isso mantendo o CoM sempre sobre o polígono de suporte da posição atual do robô. Basicamente, usando cinemática inversa, enviamos a posição da perna para a posição onde o CoM está sendo projetado, como realizado em [20]. Dessa forma, garantimos que

ele sempre está sobre o polígono de suporte.

Para isso, calculamos a posição do CoM com a eq. 2.35. Ela nos trará um vetor tridimensional. Os dois primeiros componentes nos trazem a projeção do CoM no plano xy, em relação ao torso do robô. O valor do terceiro componente do vetor depende da posição vinda do Kinect. Temos assim uma posição para onde devemos enviar o efetuador final do robô. Levando em conta que o NAO possa estar inclinado, utilizamos as informações dos sensores inerciais para encontrarmos essa rotação. E aplicamos essa rotação para corrigir a translação a ser aplicada no robô.

3.5 Conclusão

Vimos que a implementação em *software* da nossa solução explora explora a interação entre Kinect, ROS e NAOqi, além da utilização de uma biblioteca de quatérnios duais para a robótica. No capítulo a seguir, veremos os resultados do trabalho apresentado.

Capítulo 4

Resultados

4.1 Introdução

Foam realizados três experimentos simples para verificar a eficiência de cada técnica de controle escolhida (matrizes e quatérnios duais):

- Erguimento do braço direito no plano YZ, até a altura do ombro e retorno à posição inicial, na altura do quadril;
- Movimentação simultânea dos dois braços em dois tempos: o primeiro com o erguimento dos mesmos no plano YZ e em seguida, no plano XZ, ambas com retorno à posição inicial; e
- Erguimento da perna esquerda no plano YZ e retorno à posição inicial.

Para aplicarmos os mesmos parâmetros de entrada em todos os testes, utilizamos um pacote especial de ROS, o rosbag. Com ele gravamos alguns movimentos padrão (levantar braço, movimentar ambos os braços, levantar uma perna) e reproduzimos os mesmos a cada teste.

Além disso, como o controle com quatérnios duais nos permite algumas modificações de ganho e de amortecimento. Modificamos essas constantes durante o experimento, para verificar seu efeito no robô real.

4.2 Resultados

4.2.1 Movimentação do braço direito

Neste experimento, movimentamos o membro superior direito do robô conforme podemos observar na Figura 4.1.

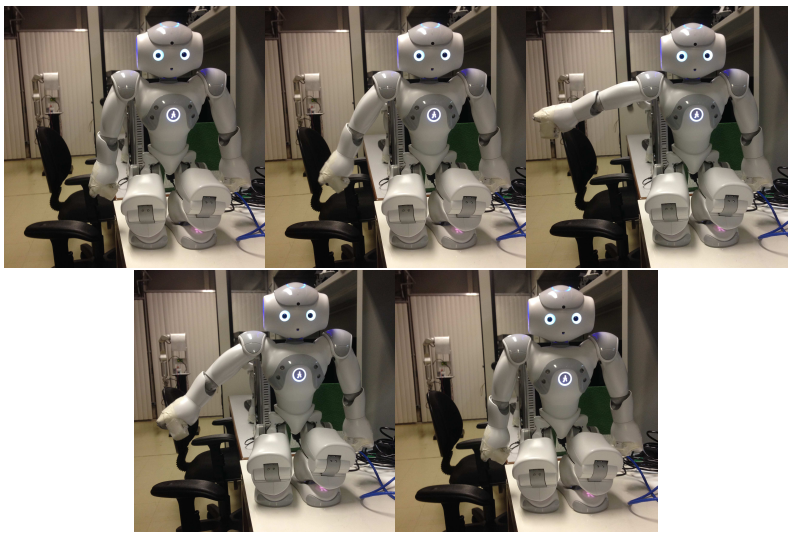


Figura 4.1: Sequência de movimentação do braço direito do NAO

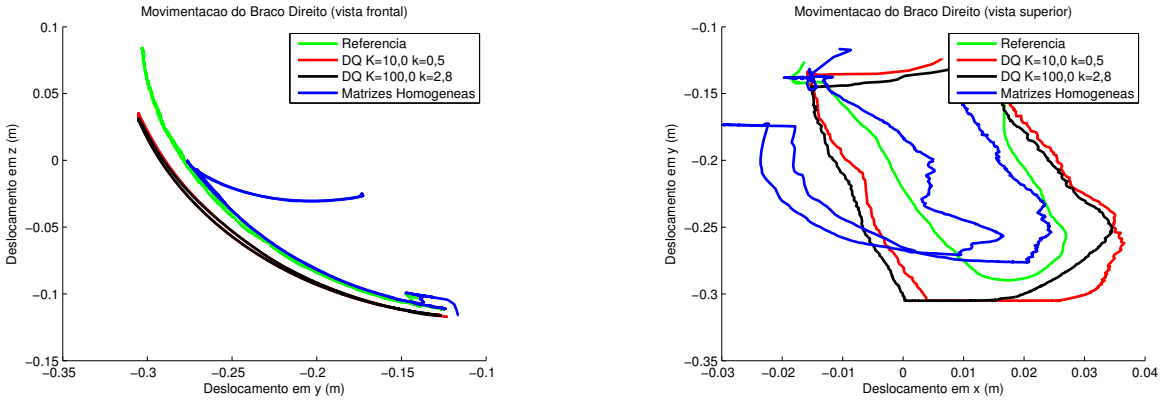


Figura 4.2: Gráficos da movimentação do braço direito com o uso de matrizes homogêneas.

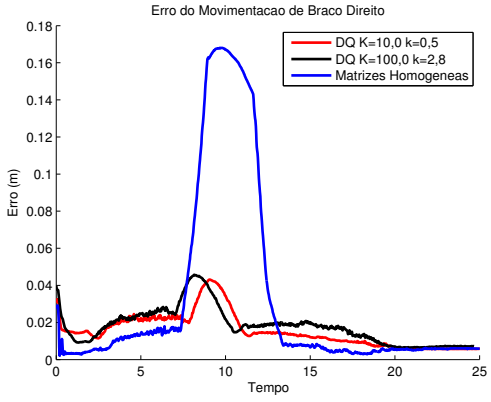


Figura 4.3: Gráfico da norma do erro de movimentação do braço direito.

Aplicando as duas abordagens de controle apresentadas nesse trabalho, adquirimos os dados de movimentação da Figura 4.2. A Figura 4.3 apresenta a norma do erro do movimento em relação à

referência. Utilizamos ainda diferentes parâmetros de ganho e de amortecimento para o controle com quatérnios duais para fins de comparação.

No gráfico da Figura 4.2 à esquerda, nota-se grande semelhança na trajetória com os diferentes parâmetros de controle com quatérnios duais. O algoritmo de controle da Aldebaran segue bem a referência até o momento em que o braço ultrapassa a origem do sistema de coordenadas do NAO (que fica na altura do peito). Nesse momento, a mão descreve um movimento errôneo que se deve provavelmente à limitação de representação espacial de um corpo rígido com matrizes. Visualmente, o código com quatérnios duais de maior ganho gera uma movimentação mais rápida no robô. Isso ocorre pois quanto maior o ganho maior deve ser a amplitude do movimento do manipulador do humanóide, de acordo com 2.34.

Na Figura 4.3, podemos analisar as normas dos erros das movimentações em relação à referência. Em geral, os erros se mantém abaixo de 0,03. Eles crescem quando a mão está chegando ao ponto mais alto. O controle com matrizes homogêneas apresenta um distanciamento pequeno à referência, porém ele é mais sujeito a singularidades, o que torna o seu uso menos confiável. No gráfico que nos apresenta a vista superior do movimento, na Figura 4.2, podemos observar que temos um resultado mais fluido e contínuo utilizando o controle com quatérnios duais.

No código em que adotamos um ganho e um amortecimento maiores, temos mais rapidez e mais suavidade no movimento.

4.2.2 Movimentação simultânea dos braços

Neste teste, o robô inicia seu movimento com os dois braços abaixados e em seguida os levanta para os lados de forma semelhante ao exercício anterior, mas agora com os dois braços. Em seguida, ele os abaixa e os ergue novamente, desta vez para a frente, sempre mantendo os membros superiores estendidos e os abaixa em seguida. Essa sequência está ilustrada na Figura 4.4.

Na Figura 4.5 podemos observar a trajetória realizada pelos braços para a movimentação desejada utilizando os diferentes algoritmos. Na primeira imagem, vemos todos os resultados juntos. Para facilitar a análise, outros 3 gráficos foram criados, um para cada estratégia comparadas à referência. A partir disso, podemos tirar algumas conclusões. O gráfico com o experimento realizado com quatérnios duais com baixo ganho, acompanha de forma satisfatória o movimento. Ele possui a vantagem da representação em quatérnios duais e o baixo ganho ao assegurar uma velocidade menor, os movimentos tendem a se aproximar mais dos desejados.

Da mesma forma que na subseção anterior, o braço direito se movimenta de forma errônea quando a altura do braço deve ultrapassar o eixo Y do sistema de coordenadas do robô. O controle com quatérnios duais oferece segurança na representação espacial dos dados.

Podemos observar os erros de movimentação dos braços na Figura 4.6 que para o braço esquerdo, o controle com matrizes e com baixo ganho executam o movimento com maior eficiência, enquanto que para o braço direito, os dois controles com quatérnios duais são os mais eficientes.

Figura 4.4: Sequência de movimentação de ambos os braços do NAO

4.2.3 Movimentação da perna esquerda

Neste teste, esperamos que o robô movimente sua perna esquerda para frente e volte como na Figura 4.7.

A Figura 4.8, nos apresenta as trajetórias seguidas pela perna aplicando-se as diferentes técnicas de controle adotadas. Vemos que no plano que nos mostra a vista superior do robô, o algoritmo com quatérnios duais de maior ganho gera um erro na movimentação da perna, fazendo que ela se movimente um pouco mais para trás do que o esperado. Na Figura 4.9 nos apresenta a norma da distancia da posição do atuador final à referência.

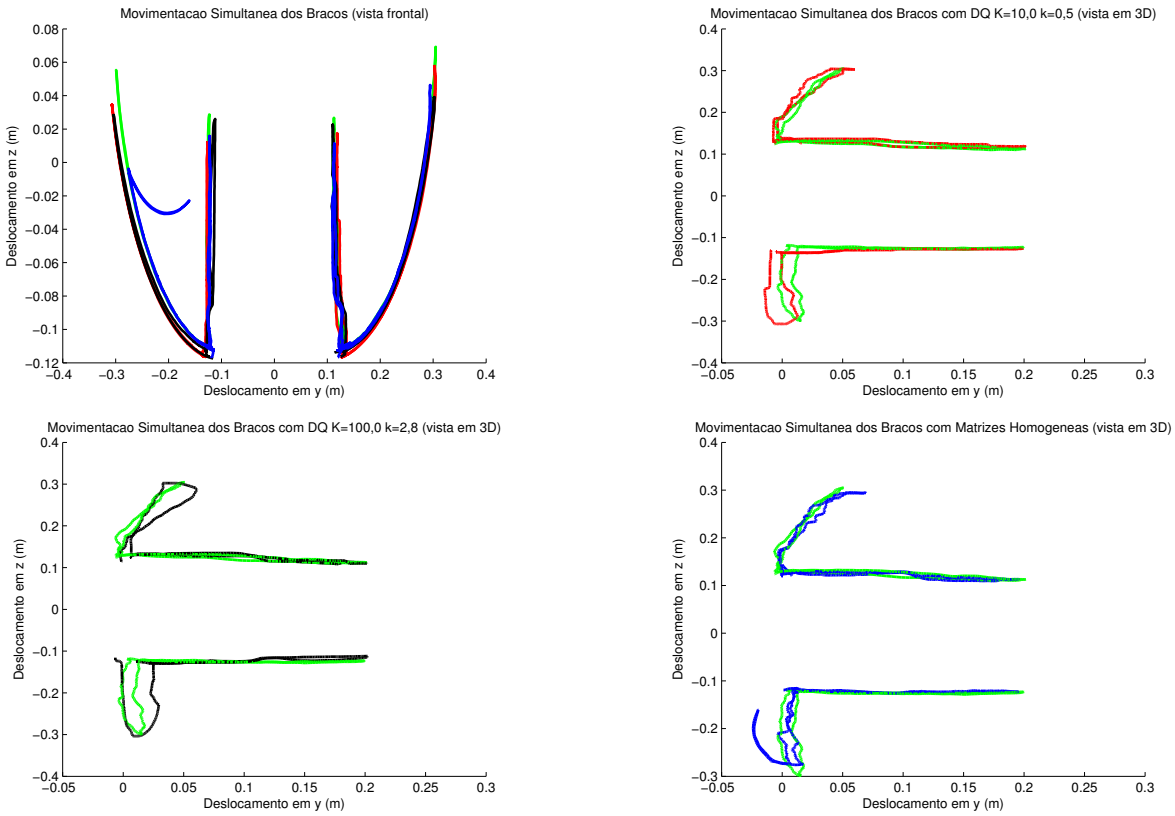


Figura 4.5: Gráficos da movimentação do simultânea dos braços.

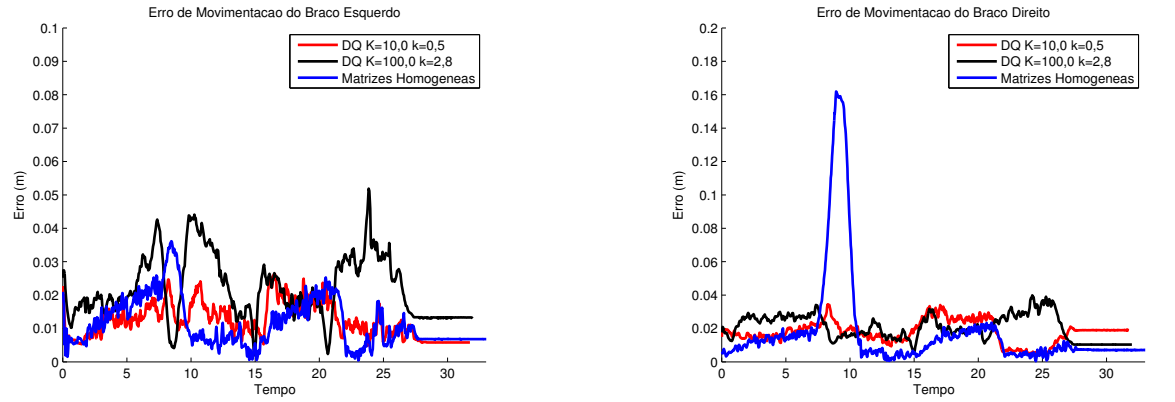


Figura 4.6: Gráfico da norma do erro de movimentação dos braços esquerdo e direito.

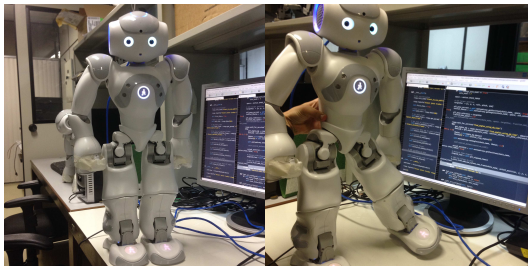


Figura 4.7: Sequência de movimentação da perna esquerda do NAO

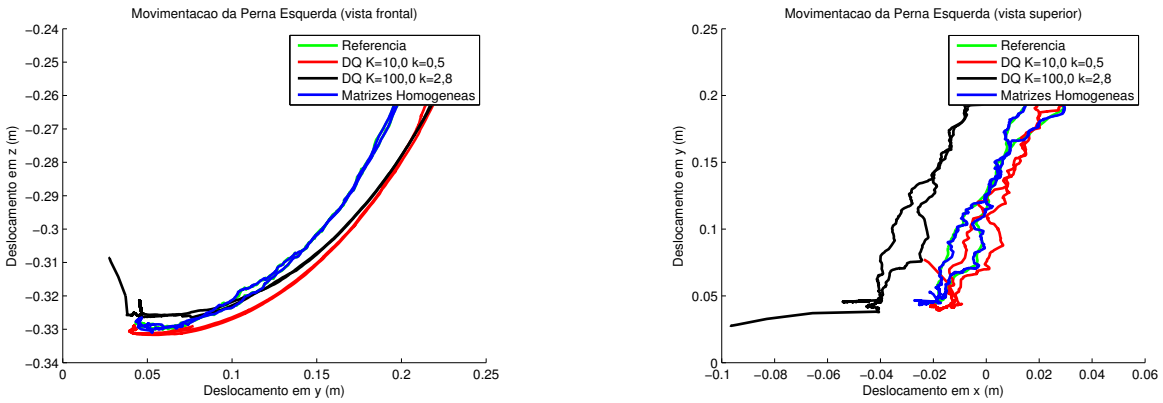


Figura 4.8: Gráficos da movimentação da perna esquerda .

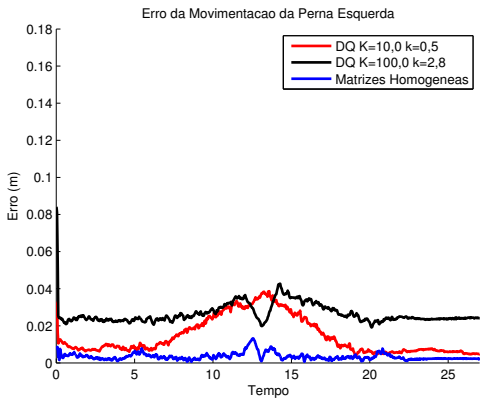


Figura 4.9: Gráfico da norma do erro de movimentação da perna esquerda.

Capítulo 5

Conclusões

O controle de corpo inteiro de um humanóide por teleoperação traz diferentes desafios. Um deles está na modelagem do robô. Vimos que, separando esse autômato em cadeias cinemáticas, poderíamos utilizar os métodos existentes para controle de manipuladores no NAO. Utilizamos então a cinemática inversa para o controle de movimentação do robô.

Toda a implementação foi realizada no ROS, um ambiente que facilita a criação e o desenvolvimento de aplicações para a robótica. Assim, para definir as novas posições dos membros a serem controlados, utilizamos o sensor Kinect, que através de um pacote do ROS nos fornece as posições de cada efetuador-final do usuário. Essas posições sofrem então uma mudança de escala e são translacionadas para encontrar a posição equivalente no sistema de coordenadas do robô. Em seguida, calculamos as posições das juntas que devem levar o sistema para a posição desejada.

Utilizamos duas técnicas diferentes para aplicar os conceitos de cinemática direta e inversa no humanóide. Os quatérnios duais se mostraram muito eficientes em definir iterativamente a solução para o problema.

Outro desafio que tentamos resolver foi na manutenção do equilíbrio do robô. O trabalho realizado nesse sentido funcionou de forma aceitável no simulador, no entanto, não no modelo real do autômato.

5.1 Perspectivas Futuras

Para trabalhos futuros, podemos melhorar os algoritmos e as técnicas para manter o equilíbrio do humanóide. Além disso, a criação de filtros que reduzam o erro e a instabilidade do sistema são importantes para sua evolução.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CHELLALI, R. *Tele-operation and Human Robots Interactions*. 2010. Disponível em: <http://cdn.intechopen.com/pdfs/10558.pdf>.

[2] STANTON, C.; BOGDANOVYCH1, A.; RATANASENA, E. Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning. *Proceedings of Australasian Conference on Robotics and Automation*, 2012.

[3] KOENEMANN, J.; BURGET, F.; BENNEWITZ, M. Real-time imitation of human whole-body motions by humanoids. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014.

[4] ZUHER, F.; ROMERO, R. Recognition of human motions for imitation and control of a humanoid robot. *Brazilian Robotics Symposium and Latin American Robotics Symposium*, 2012.

[5] MURRAY, R. M.; LI, Z.; SASTRY, S. S. *A Mathematical Introduction to Robotic Manipulation*. [S.l.]: CRC Press, 1994.

[6] HAMILTON, W. R. *Elements of Quaternions*. [S.l.]: Green Longmans, 1969.

[7] WU, Y. et al. Strapdown inertial navigation system algorithms based on dual quaternions. *Transactions on Aerospace and Electronic Systems*, pp.41, p. 110 – 132, 2005.

[8] CLIFFORD. Preliminary sketch of biquaternions. *Oxford Journals*, pp., p. 381–395, 1873.

[9] TRAUTMAN, A. Clifford algebras and their representations. In: FRANÃ§OISE, G. N. J.-P.; S.T, T. (Ed.). *Encyclopedia of Mathematical Physics*. [S.l.]: Elsevier, 2006. p. 518–530.

[10] HAN, D.-P.; WEI, Q.; LI, Z.-X. Kinematic control of free rigid bodies using dual quaternions. *International Journal of Automation and Computing*, pp., p. 319–324, 2008.

[11] SICILIANO, B. et al. *Robotics - Modelling, Planning and Control*. [S.l.]: Springer, 2009.

[12] DENAVIT, J.; HARTENBERG, R. S. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of Applied Mechanics*, pp.22, p. 215–221, 1955.

[13] PAUL, R. *Robot manipulators: mathematics, programming, and control : the computer control of robot manipulators*. [S.l.]: MIT Press, 1981.

[14] WHITNEY, D. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, pp.10, p. 47–53, 1969.

[15] DOMBRE, E.; KHALIL, W. *Robot Manipulators - Modelling, Performance Analysis and Control*. [S.l.]: Iste, 2007.

[16] BUSS, S. R. *Introduction to inverse kinematics with jacobian transpose pseudoinverse and damped least squares methods*. out. 2009. Disponível em: <http://math.ucsd.edu/ sbuss/ResearchWeb/ikmethods/iksurvey.pdf>.

[17] ADORNO, B. V. *Two-arm manipulation: from manipulators to enhanced human-robot collaboration*. Tese (Doutorado) — Universite Montpellier 2, 2011.

[18] KILNER, C.; COLLETTE, C.; GOUAILLIER, D. *ALMath - an optimized mathematic toolbox for robotics*. 2008–2011. Disponível em: <https://community.aldebaran.com/doc/1-14/ref/libalmath/overview.html>.

[19] MARINHO, M. M. et al. Manipulator control based on the dual quaternion framework for intuitive teleoperation using kinect. *Proceedings of the 2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*, p. 319–324, 2012.

[20] KOFINAS, N. *Forward and Inverse Kinematics for the NAO Humanoid Robot*. 2012. Disponível em: <www.intelligence.tuc.gr/lib/downloadfile.php?id=430>.

ANEXOS